Analyzing Phenotypes in High-Content Screening with Machine Learning

by

Lee Zamparo

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Analyzing Phenotypes in High-Content Screening with Machine Learning

Lee Zamparo
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2015

High-content screening (HCS) uses computational analysis on large collections of unlabeled biological image data to make discoveries in cell biology. While biological images have historically been analyzed by inspection, advances in the automation of sample preparation and delivery, coupled with advances in microscopy and data storage, have resulted in a massive increase in both the number and resolution of images produced per study. These advances have facilitated genome-scale imaging studies, which are increasingly frequent. Although the sheer volume of data involved strongly favours computational analysis, many assays continue to be scored by eye. As a scoring method, visual inspection limits the rate at which data may be analyzed, at increased cost and decreased reproducibility. In this thesis, we propose computational methods for data analysis of HCS data.

We begin with feature data derived from confocal microscopy fluorescence images of yeast cell populations. We use machine learning methods trained on a small labeled subset of that feature data to robustly score each population with respect to a DNA damage focus phenotype. We then introduce a method for using deep autoencoders trained using a label-free objective to perform dimensionality reduction. This allows us to model the non-linear relations between features in high-dimensional data. The computational complexity of our approach scales linearly with the number of examples, allowing us to train on a much larger number of samples. Finally, we propose an outlier detection method for discovering populations that present significantly different distributions of cellular phenotypes as compared to wild-type using nonparametric Bayesian clustering on the low-dimensional data. We evaluate our methods against comparable alternatives and show that they either meet or exceed the level of top performers.

Acknowledgements

To begin, I thank Zhaolei Zhang for being my advisor. At times, this has entailed being a mentor, facilitator, or supportive voice. I also thank Alan Moses and Quaid Morris for serving on my advisory committee. Their encouragement, suggestions, and criticism have helped me become a better researcher and have improved this thesis.

I thank my truly top-class collaborators on the marker project, Erin Styles and Karen Founk, for bringing me on as a partner. As though by some process of alchemy, their dedication, organization, and competency transformed our shared hard work into pure enjoyment. I also thank my excellent colleagues past and present: Jingjing Li, Renqiang Min, Anastasia Baryshnikova, Ruth Isserlin, Gerald Quon, Recep Colak, Brian Law, Amit Deshwar, Iain Wallace and Leopold Parts. Some people say the best learning in graduate school comes from your peers; I believe them. I want to thank my parents, for without their guidance and support I would not never have had the ability or fortitude to persevere through life's challenges. Finally, I thank Vickie, who has shown me how truly happy two people can be when each cares deeply about the other. I will always strive to repay that gift in kind.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Fluorescence microscopy has become a robust quantitative method for cell biology. Thanks to improvements in both instruments and computational tools, genome-scale imaging studies are becoming increasingly common. Genome-scale imaging may be used in a number of different applications, such as measuring the effects of drugs on cells [92], determining the proteome wide localization patterns [25], tracking the movement of cellular structures [77], and mapping the gene expression in developing embryos [121] and adult brains [91, 90].

It may also be used to measure the effect of genetic perturbations on cell morphology. Since cells do not respond homogeneously to stimuli, the ability to measure single cell responses is critical to the identification of different sub-populations. In this thesis, we refer to the automated capture and quantitative analysis of fluorescently labeled cellular images as high-content screening (HCS). The increased throughput characteristic of HCS experiments is due to automation of sample handling and microscopy; the development of robotic controlled stage positioning, fluorescence filters, camera acquisition and auto-focusing have all increased the throughput potential of HCS studies. Taken together, these advancements open up the possibility of quantitative morphological assessment on the genome-scale for cell populations bearing specific mutations or treatments. HCS has been effectively used to study a wide ranging collection of topics including organelle morphology [63, 126], drug discovery [75], signaling pathways [6], sub-cellular protein localization [88, 81, 46], and functional genomics [38].

Another contributing factor to the increased throughput of HCS experiments has been the development of software to design and carry out pipelines of image analysis tasks that arise in HCS, such as CellProfiler [20], CellHTS [18] and the Open Microscope Environment [115]. These frameworks allow scientists to identify objects of interest in their images, then to define and measure functions of the observed fluorescence in the objects as features, without requiring expertise in image analysis. These advances in HCS automation have both reduced its cost, and broadened its scope. Yet much of the subsequent work, the data analysis, remains as a bottleneck. Even in some contemporary studies, exploratory data analysis of microscopy images is undertaken by eye, and the data analyzed by hand [30]. This is prohibitively time consuming, error prone and difficult to reproduce. Recent applications of large-scale machine learning in HCS, either directly to the images or to image features previously extracted, have resulted in impressive advances in the scale and granularity of conclusions that can be drawn from HCS data [46, 36, 93, 38]. While these selected successes are encouraging, there remains a need for more methods that provide in-depth exploratory data analysis and that can succeed in a scenario where access to labeled data is limited.

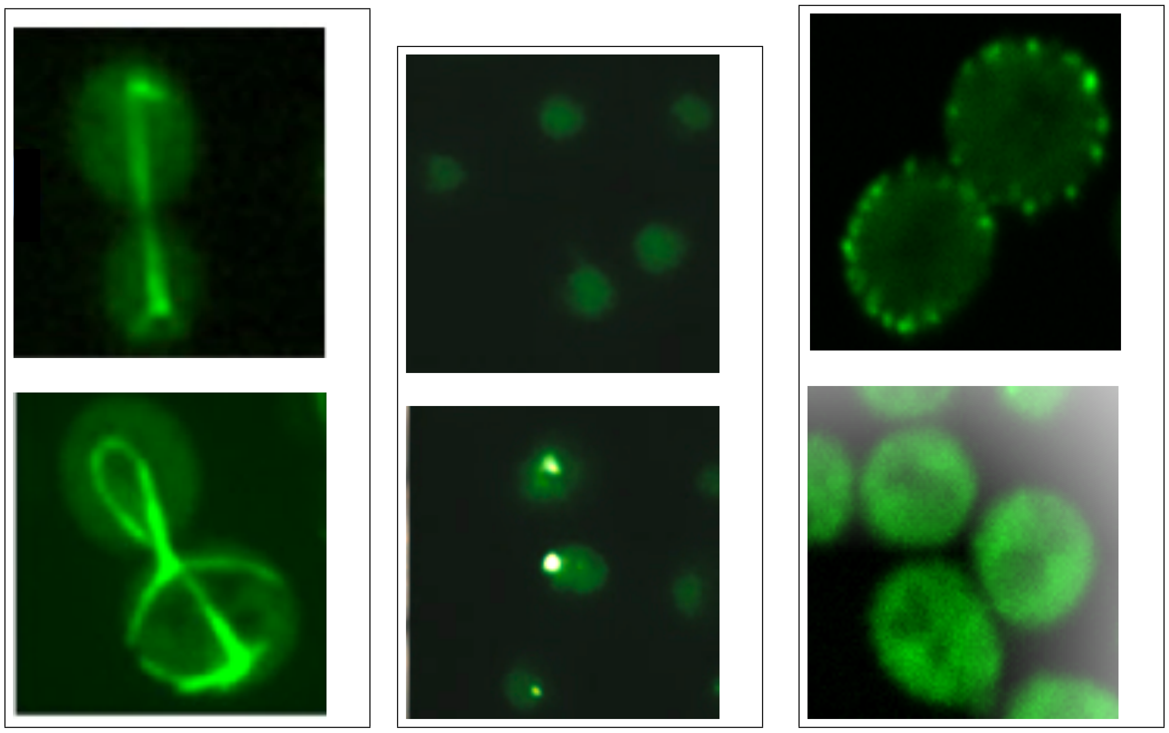


Figure 1.1: Example images from HCS studies of yeast. Each panel depicts a pair of cells from studies that used GFP fusion proteins to reveal the effects of genetic deletions on sub-cellular morphology or protein localization. Left panel: (top) a sample wild-type yeast cell with normal GFP-Tub1p marker, (bottom) an *slk19*Δ yeast cell displaying abnormal spindle pole formation. Image is courtesy of Franco Vizeacoumar. Middle panel: (top) wild-type yeast cells with Rad52-GFP marker, (bottom) an *irc6*Δ sample yeast cell displaying DNA damage foci. Image is courtesy of Erin Styles. Right panel: (top) wild-type yeast cells with Sla1-GFP marker, (bottom) an *end3*Δ sample yeast cell displaying a diffuse actin distribution. Image is courtesy of Mojca Mattiazzi.

1.1 Different problems in microscopy imaging

The image analysis problems that are common in HCS, such as finding and recognizing biological objects, occur within a greater context of image labeling and object recognition, which are central problems in computer vision [64]. However, there are several characteristics specific to microscopy imaging that make these problems sufficiently different from object recognition or scene labeling in natural images. Below we split these characteristics into those that simplify and those that complicate object recognition.

1.1.1 Simplifications in microscopy imaging

The first characteristic that simplifies object recognition is the use of a fixed camera position relative to the cell objects, so there is no variation in viewing angle (or pose) across images. Images in confocal microscopy are lit from a laser whose position relative to the image plane is also fixed. By using different lasers for each colour channel, confocal images can reduce fluorophore cross-talk, and thus minimize the variation in lighting conditions. The scale of objects in the images is near constant. We say "near constant" because there is some variation introduced in choosing which point on the Z-axis is used to maximize the number of in-focus cells. Thus, choosing to use one image along the Z-axis which will change image to image may introduce noise in the form of cell size. While three-dimensional imaging is possible with confocal microscopy, it introduces a multiple image alignment (or registration) problem that is difficult to solve.

1.1.2 Complications in microscopy imaging

One complication of microscopy imaging is that the process of sample preparation perturbs the objects, often causing harm or death. In fixed cell imaging, samples are treated with formaldehyde to preserve protein contacts[1]; however, this may cause cell shrinkage or vesiculation in membranes [113]. In live cell imaging, excessive or prolonged exposure to light will damage cells by phototoxicity. Another complication of longer illumination periods is the diminishing excitation response of fluorophores due to photobleaching [68]. This danger is amplified if cells are imaged in three dimensions using conventional confocal microscopy. While advances in microscopy, such as stimulated emission depletion microscopy [108] and light-sheet microscopy [116], can mitigate these dangers, these technologies are not widespread. Therefore in live-cell confocal imaging, there exists a trade-off between image quality (which is higher when light-excitation dose is high) and photobleaching, which can reduce image quality due to a shorter excitation period and thus to a lower signal-to-noise ratio [34]. Finally, in confocal microscopy images, every point observed in the image is assumed to be convolved with a point spread function (PSF). Calibration of PSF deconvolution algorithms are usually performed by software provided with the microscope apparatus, using theoretical measurements specific to the model, or using imaging calibration beads of known shape and size. However, what is not accounted for is how the light passing through the cell specimens will perturb the PSF, leading to additional noise due to out-of-focus light.

1.2 Formulating HCS image interpretation as a computational problem

HCS image interpretation, when framed as a problem in machine learning, is usually formulated as an object labeling task. Given a set of objects from an image containing cells (shown in Figure 1.1), the goal is to produce an output label for each object that reflects the phenotype. Object recognition[2] is a mature sub-field of computer vision. We have previously noted that cell images have distinct characteristics; while objects appear in a constant scale and pose, they are illuminated with less light and are noisy. In addition, there are other factors that complicate the direct application of object recognition algorithms. First, algorithms for object recognition are usually trained on sets with labels, such as NORB, ImageNet, TinyImages or CIFAR-10, where examples for each class are abundant [64, 123, 114, 61]. This is often not the case in HCS for three reasons. The different phenotypes displayed by cell objects are often subtle and not easy to label except by domain experts. Second, the process of culturing cells, applying a treatment, and then capturing images by microscopy is costly and time consuming. Finally, the sheer multitude of phenotypes possible to investigate via HCS means that public data for many given phenotypes are lacking.

[1]this procedure is often referred to as cross-linking
[2]the task of identifying objects within a larger image and classifying these objects

Often the objects represented by class labels in many recognition tasks are found in everyday life. Due to their ubiquity, they may be easily described and recognized. For images containing such objects, labels can be readily applied by lay people. This is not the case for cell objects in HCS images where phenotypes may be subtle, and their description may be ambiguous. Object labels must therefore be applied by human experts possessing sufficient domain knowledge to recognize them. This dependence on human expertise makes labeled data expensive to acquire. Another difference in HCS object recognition is that the inherent biological variability among cell populations often leads to observing a variety of phenotypes, even in genetically identical populations. In practice, this further complicates the labeling process, especially for phenotypes that have exceptionally low penetrance, making them difficult to detect. One advantage of high-throughput experiments is that the amount of unlabeled data is usually very large. Methods that are able to learn using strictly unlabeled data should be advantaged over those restricted to labeled data.

1.3 Aims and organization

The goal of this thesis is to develop new machine learning methods for HCS data analysis, which are tailored to suit its specific conditions and challenges. In particular, this thesis focuses on the following three problems that arise when applying machine learning in data analysis for HCS experiments:

Robust classification and scoring of populations: How do we leverage a small amount of labeled training data to train a classifier that can be applied across different conditions and replicates? We propose a method and procedures for evaluating our model. We develop summary statistics for each population; finally, we incorporate results from different replicates and experimental conditions to increase our power to suggest new roles for genes identified in a study of DNA damage.

Leveraging huge amounts of unlabeled data: One of the advantages of HCS is the large amount of unlabeled data. We would like to use that unlabeled data to learn a model of how different populations are structured and investigate those that differ greatly from our control populations. A natural first step is to reduce the dimensionality before trying to discover structure in the data. This is complicated by large amounts of high-dimensional data, and we show reason to suspect linear methods are too inexact. Our solution is to compress the data by constructing a deep autoencoder model that is trained to compute low-dimensional codes from high-dimensional inputs that respect complexities in relationships among covariates. We evaluate our model in relation to several comparators on a test that measures proximity in a lower dimensional space for data bearing withheld labels.

Finding populations that differ significantly from wild-type: A common goal of many HCS studies is to find populations or treatments that differ in some significant way from the wild-type or control population [88]. This differs from population scoring in two ways. First, we do not assume that there is a training set with labeled cells, so classification is not applicable. Second, we do not assume to know how many classes of cells are present in the data, only that there are some classes which appear in different proportions among the separate populations of isogenic cells, which we would like to use to characterize the populations. Describing different cell populations has been approached as a clustering problem by several groups [112, 29, 36, 46]. Building on the results of the previous chapter, we explore how perform outlier detection on the reduced data. We use sampling to construct a reference model of cell variability that adapts to the complexity of the dataset using Bayesian nonparametrics. We show how to use this model, a Dirichlet process mixture model, to characterize each population and find outliers that represent leads that can be scrutinized with more directed experiments.

The main contributions of this thesis are three methods, each of which addresses a different challenge in the analysis of HCS data. The first maximizes the utility of a small amount of labeled training data, building a model to robustly combine predictions across biological replicates. The second allows for non-linear dimensionality reduction to be applied on large unlabeled data sets, in order to better understand the structure of the data. The third uses low-dimensional data sourced from the previous contribution to characterize populations and identify those that are outliers with respect to a reference set. This enables the prioritization of resources to the most promising leads, which can then be further understood with follow-up experiments. Together, these chapters present a road-map for exploratory analysis in HCS that overcome two common problems: how to build models that best exploit small amounts of labeled data for drawing sound conclusions from genome-wide screens, and how to incorporate large amounts of (possibly high dimensional) unlabeled data for this same goal. These contributions result in new tools to mitigate the complexity and time investment required by human experts in the analysis of HCS data.

The rest of the thesis is organized as follows:

- Chapter two presents a brief overview of concepts in yeast genetics, fluorescence microscopy and HCS. Machine learning algorithms that are employed or extended in this thesis are also reviewed.
- Chapter three describes a beginning-to-end study of an HCS assay in yeast, specifically looking at the phenotype of DNA damage foci. It reviews the main pathways by which eukaryotic cells detect and repair double-stranded breaks, which should solidify the concepts of HCS, and describes the data produced. The contributions in this chapter are a pipeline of methods that robustly quantify those genetic deletions that show signs of having a role in double-stranded DNA break repair. This chapter casts HCS as a supervised learning problem, given that the pipeline uses a small labeled training set of cells.
- Chapter four details a method to construct a deep autoencoder for performing flexible dimensionality reduction on very large data sets, where non-linear relationships between covariates rule out the application of PCA. A deep neural network model is trained to learn a function that takes cell features as input and outputs a low dimensional code, effectively compressing the representation.
- Chapter five shows how low dimensional representations from the previous chapter are used to sample a control or representative model from the data set, which is used to fit a Dirichlet process mixture model. We use the reference model to evaluate the aggregated posterior responsibilities of each individual population; we use the divergence between these and the reference population as a method to identify outliers.
- Chapter six summarizes the most important findings from each chapter and discusses possible extensions for the methods so that they might accommodate more complicated experimental designs.

Chapter 2

Background

This chapter presents background material upon which work presented in the subsequent chapters of this thesis is built. We start by reviewing the experimental model and workflow for HCS, to give the reader an idea of how data from HCS experiments is collected. With the process of image generation in hand, we next describe the fundamental image analysis step of scanning each image to estimate the positions of objects within, called segmentation. We continue with a discussion of how to transform these objects, which are initially represented as a set of pixels, into feature vectors. We follow by changing gears to discuss some fundamental problems and algorithms in machine learning. We begin with dimensionality reduction, which are methods that transform high dimensional data to embed it in a lower dimensional space. We follow with supervised learning, where models learn from labeled examples to classify unlabeled input, and describe some commonly used algorithms that fall under this framework. We conclude with a discussion of unsupervised learning, where models learn some structure that is present in the input data, without any labeled examples for guidance.

2.1 High-content screening

The emergence of systems biology, which seeks to identify all the components of a given process and to understand their interaction [47], has been facilitated by high-throughput methodologies. Driven by technological advances, fast and reliable measurements of gene expression, protein expression, protein interaction, and gene regulation are now commonplace. Each of these high-throughput techniques produce a puzzle, from which a greater understanding of how the pieces combine to carry out the underlying biology. One deficiency common to these techniques is that they either provide a snap-shot of the cell state at measurement time, or a steady-state average over many samples[1]. Yet genes or proteins carry out their functions inside living cells. Protein interactions occur within close proximity and within cellular substructures[78]. High-content screeing, via automated fluoresence microscopy, provides one mechanism for systems biology to capture and query spatial and temporal patterns in populations of single cells.

High-content screening (abbrv. HCS) describes a mode of analysis based on microscopy experiments that produce large volumes of biological images. High-content imaging combines techniques from different fields, such as image processing and machine learning, to extract objects of interest from biological

[1]with the possible exception of single cell RNA seq.

images, and to subsequently analyze data derived from these collections of objects on a population level, to produce biologically meaningful conclusions [110]. These methods bear several different names, depending on the context from which they arose. A recently published survey by Peng, which took a computational perspective, refers to the field as Bioimage analysis [89]. Another computationally focused survey by Shariff et al. uses the term High Content Analysis [110]. More biologically focused papers tend to use the term HCS [65, 41, 76]. What these all share are the method of introducing genetic or chemical perturbations into a large number of isogenic sub-populations, either of cells or whole organisms, and the use of microscopy as the means of observation. Typically the perturbations are designed to investigate some aspect of cellular morphology, organization, gene expression, or protein interaction. The most common way of measuring the effect is by the application of fluorescent labeling agents, followed by imaging the cells (or organisms) as they respond to different genetic or chemical stimuli. Examples include drug lead discovery studies [76], targeted gene function assays [6], and genome wide profiling for genetic interaction studies [126].

This chapter surveys the most common components of a high content imaging study, with a particular focus on algorithms for analyzing the resulting data. Throughout this review, the terms *high content imaging*, and *high content screening* will be used interchangeably. The focus will be on the study of protein targets. For those interested in HCS for *in vivo* mRNA, Arjun et al. have published a recent review of fluorescent in-situ hybridization [94]. The typical HCS experiment can be visualized as a pipeline, with the input of each stage using the output of the previous. The stages are visualized below here in Figure 2.1.

2.2 Image acquisition

Once the samples are prepared and plated, the image capture stage can begin. When considering the different types of HCS experiments, many factors must be weighted and considered: the type of microscopy, the number of imaging channels, whether to capture 2D or 3D images, and whether to do one time point or a time series. In [65][89], the factors to consider for image acquisition include wide field versus confocal microscopy, excitation source (lamp versus laser), magnification objectives, filters and detectors for light separation, auto-focus mechanisms, environmental control, and throughput capacity. Most of these technical details are beyond the scope of this review, but they are listed here so readers have an idea of how much work and precision underlies the process of biological image generation.

2.2.1 Microscopy

The most common type of microscopy used for HCS is fluorescence, which offers a good trade between resolution and experimental flexibility. Samples can be imaged in a wide variety of conditions, either live or fixed. Schermelleh at al. make the analogy of a capturing images with a fluorescence microscope to painting with a fuzzy brush [108]: the shapes of each object in the specimen, represented as a distribution of intensity values, are determined by what is called the point spread function (PSF) of the brush, since it describes how the light from one point on the object is spread in the image. This process of 'fuzzy brush' painting can be viewed mathematically as a convolution of the object with the PSF, meaning that the level of fine detail in the image depends upon the PSF of the microscope. Modern confocal laser scanning microscopes (CLSM) can resolve images to distinguish objects down to a level of 200 to 350nm margin of separation, which is enough to investigate the spatial distribution and dynamics

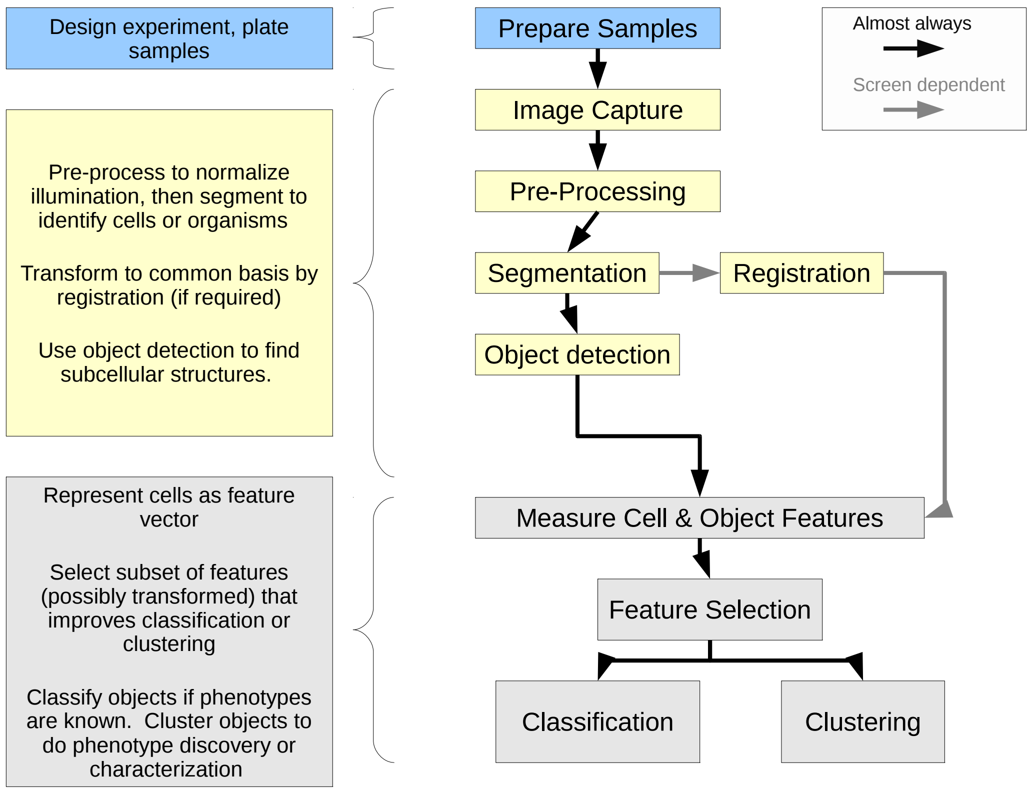

Design experiment, plate samples
Prepare Samples
Almost always
Screen dependent
Pre-process to normalize illumination, then segment to identify cells or organisms
Transform to common basis by registration (if required)
Use object detection to find subcellular structures.
Image Capture
Pre-Processing
Segmentation
Registration
Object detection
Represent cells as feature vector
Select subset of features (possibly transformed) that improves classification or clustering
Classify objects if phenotypes are known. Cluster objects to do phenotype discovery or characterization
Measure Cell & Object Features
Feature Selection
Classification
Clustering

Figure 2.1: A sample HCS pipeline for a cell based assay. The arrows indicate which stages are almost always present, and which are dependent on the type of experiment. Note that often the bottom two sections (image analysis, data analysis) will be re-examined or repeated as ideas or results dictate.

of many sub-cellular structures [108]. However, many important processes such as signaling, protein complex formation or protein-DNA binding, occupy a much smaller range of tens to few hundred nm. This is beyond the reach of conventional light microscopy. Given perfect lenses, optical alignment and aperture size, the optical resolution of light microscopy is still limited by diffraction to half of the wavelength of the light source used. One solution to this problem is electron microscopy (EM), which by virtue of using electrons rather than photons uses 10^5 times smaller wavelength, resulting in up to 100 times greater resolving power [108]. This increased resolution comes at a cost that rules out its use for high throughput screening: operating costs are much greater, experimental flexibility is limited by the strict environmental requirements, and the rate at which samples can be imaged is much lower. Below I will summarize the properties of both wide-field and confocal microscopy, standard methods for fluorescence based microscopy, as well as briefly touch upon three emerging techniques that offer greater resolution. For more involved discussions of the optical theory underlying confocal microscopes, see the book chapter by Smith [113], which also includes detailed advice for microscope configuration and experimental conditions. Schermelleh et al. recently published a guide to super resolution fluorescence microscopy [108]. These techniques are also referred to in a review by Megason et al. [78], but not described. We shall cover only the confocal microscopy in detail here, since the data used in later chapters of this thesis were drived from automated confocal microscopy.

Confocal

Confocal microscopy has two major improvements over the wide field microscope: laser light illumination which can be more precisely focused, and a confocal pinhole that prevents excited fluorophores that are not in the focal plane from reaching the detector [65]. Several types of confocal microscopes are available. The most common type is the confocal laser scanning microscope (CLSM), which captures images by scanning the specimen with a focused beam of light from a laser and collecting the emitted fluorescence signals with a detector. This provides greater resolution, but comes at a cost that the field of view must be raster-scanned point by point to form the image. Many CLSMs are equipped with different lasers: an argon laser (488nm) with a green helium-neon (HeNe) laser (543 or 594nm) and a red HeNe laser (633nm). The argon laser can excite either cyan or yellow variants of GFP (CFP and YFP), which the green and red lasers excite GFP and RFP respectively. Two additions to conventional CLSMs that help increase the image acquisition rate are slit-scanning and disk-scanning methods. Slit scanning CLSMs have a horizontal laser used to excite the field on the Y axis, with a fine gated slit blocking out of focus light, allowing the specimen to be line-scanned rather than spot scanned. Disk-scanning CSLMs use a spinning disk that contains many fine pinholes that allow light to pass through and excite the specimen in the focal plane. The disk revolves rapidly (1000 to 5000 rpm), causing the spots to cover the specimen as uniformly spaced scan lines. Fluorescence emitted by the specimen returns through the same pinholes in the disk that provided the excitation light. Since the point light sources and detector pinholes combine to block out of focus fluorescence, the higher resolution of CSLM is preserved and acquisition rates are increased to handle live cell imaging (approximately 30 fps, versus 1 fps for spot scanning). Problems with both spinning disk and line scanning CLSM include decreased illumination to the specimen from light loss through the pinholes or slit. In addition, since the pinhole sizes are fixed, optimal confocal resolution can only be achieved for one objective magnification [113].

Illumination correction

Illumination correction is a pre-processing step where artifacts from uneven illumination of images that are generated from the microscope are dealt with. Typically this involves ensuring that the illumination for each image is standardized across wells on a plate, and across all plates in the study. In Ljosa and Carpenter [72], they fit a smooth illumination function to the image, and each pixel intensity is adjusted by dividing by the estimated value in the illumination function at that position. For objects at the periphery of the image, where illumination may be uneven or dim, this could cause bright objects to appear dimmer, and lead to false negative errors in subsequent classification. Shariff et al. [110] also recommend this pixel level pre-processing, either by the illumination function method, or by contrast stretching. Contrast stretching is where the pixel intensity values are re-normalized to lie within a given interval.

2.3 Image analysis

The purpose of the operations described in this section is to take as input images, identify and extract those regions that are biologically interesting, track or trace them across time if required, and measure different aspects of their intensity and shape to represent them as vectors of feature values. These techniques, segmentation, registration, tracing and tracking are drawn from the field of computer vision. In this section, I will review the problems of detecting meaningful objects in an image (segmentation), as well as the problem of aligning a set of images (registration). There are many tool-kits and software suites that offer different alternatives for these tasks, such as ITK (The Insight Segmentation and Registration Toolkit, www.itk.org), CellProfiler [58], the Matlab ImageProcessing toolbox (http://www.mathworks.com/help/toolbox/images/), and ImageJ [2].

2.3.1 Segmentation

Segmentation for HCS usually is posed as the process of identifying biologically relevant objects from an image field. It is one of the most important steps in the HCS pipeline, and usually determines the degree to which subsequent steps can produce meaningful results. Poor segmentation is a huge source of both type I and type II errors in downstream analysis (miss good objects, pick up dust, contamination, ghost objects). Ljosa and Carpenter [72] break down segmentation into methods that are either global or local. Global methods act on the whole image. They review two such methods. The first is a simple process that fits a two-class mixture model on log transformed pixel intensity to learn an intensity based threshold separating foreground from background. They also include a method proposed by Otsu which is a parametrized search over the threshold parameter values that minimizes the weighted sum of intensity variance within the two classes. The local methods they review are also threshold based, but involve setting different thresholds to each local neighbourhood of the image. Both these methods would be subject to a second stage where the object borders are refined. Peng cites low fluorescence signal

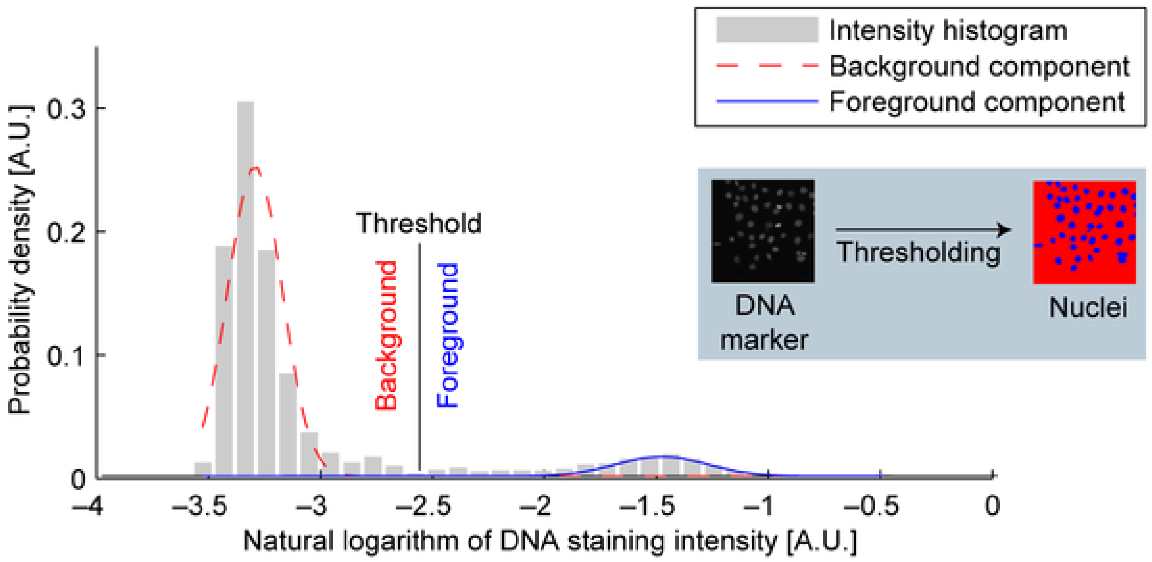


Figure 2.2: Reproduced from Ljosa and Carpenter [72], this histogram of intensity values depicts how mixture modeling can be used for a basic segmentation model to separate foreground objects from the background of the image.

of genuine objects relative to background noise illumination, and variability in the shapes of objects as the main complications for segmentation[89]. The review also claims that the success of segmentation depends upon the selection of features appropriate for identifying the objects of interest. For example, chromatin is well identified by texture features, where as nuclei are identified well by shape features. Models surveyed here include globular-template (i.e circle or ellipse) based segmentation, Gaussian mixture modeling, and active contour methods. Peng distinguishes methods for segmenting non-round cells (e.g neuronal cells) which include first applying watershed segmentation, then post-processing with a method based on a model specific to the cell type. Coelho et al. [24] describe only seed based methods. These are methods that take as input the array of pixels for the image, as well as an initial set of pixels that each identify a portion of an object. The first is Voronoi segmentation, where each pixel is assigned to the nearest seed based on some distance measure between pixels and seed points. The second is seeded watershed. In this algorithm, the pixel intensity values are inverted, and the image is modeled as a landscape. The seeds represent basins in the landscape, which are flooded. When two adjacent flooded basins are about to merge, a dam is built that represents the boundary between the two cells. Shariff et al. mention Voronoi segmentation, model-based methods, seeded watershed, active contour-based approaches, graphical model segmentation [110]. They characterize most methods as two stage methods: the first stage generates a coarse segmentation, which is refined in the second stage. The cite as examples Voronoi, seeded watershed, and active contour methods. These methods begin by defining a window that contains the cell. The coarse boundary is deformed iteratively until it is refined to match the boundary of the cell. From a survey of the published literature, it seems that simple two-stage methods are common. One example is Fuchs et al. [38], who performed two step segmentation to identify HeLa cells. First they segmented nuclei by adaptive thresholding, distance gradient transform and watershed to identify the nuclear membrane. Then using the nuclei as seed regions, they used Voronoi segmentation to identify the cell borders. There are alternatives to segmentation. In [95], Ranzato et al. present a general purpose method of classifying biological particles. They employ an object detection method that finds "interest points", and establishes a bounding box (or circle) that best encloses the object. Their subsequent analysis computes features based on all the pixels in the box, which includes both background pixels as well as objects.

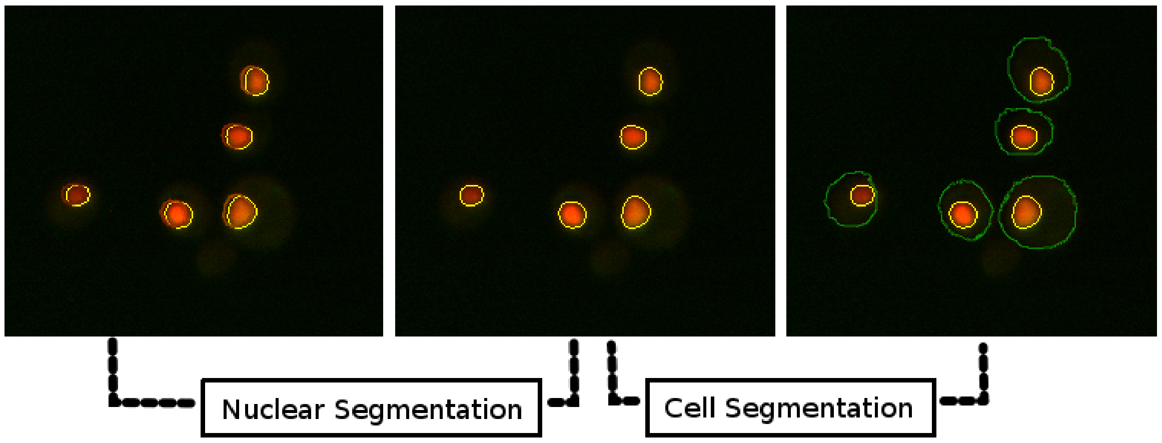


Figure 2.3: A two stage process for cell segmentation [67]. The leftmost panel shows a cropped region of a field of view image in a plate of yeast cells. The middle panel shows the same cells with the nuclei segmented by watershed with an intensity gradient transform. The rightmost panel shows the segmented cell borders, where the used as seed regions for watershed segmentation to identify cells.

2.4 Representation of images as features

Once objects are segmented from the background of the images, often times they are re-represented as collections of features. Sometimes this choice is enforced by computational tractability: large objects represented as pixels roughly take the square of their lengths to store, multiplied by the number of light channels. This leads to the problem of how to choose the type and amounts of features to represent each object. Many studies choose to measure a very large number of these features for each of the objects or images in their study, as often it is not known which features will be most useful when analyzing the data. This excess of features complicates data analysis, and imposes a constraint on methods which are computationally intensive with respect to the number of features. There are two common solutions to this problem. The first is to reduce the number of features under consideration by selecting some subset of informative features. The second is to learn a map from the high dimensional feature space into a much lower dimensional feature space that preserves the information and relationships as faithfully as possible. Feature selection is a problem that we will not review here, as it falls beyond the scope of this work and is well reviewed elsewhere [43]. Instead, we will describe several methods for dimensionality reduction in section 2.5.1. For images of cells and sub-cellular compartments, pixel based objects are measured using different features to achieve a more tractable representation. Features usually fall into one of three broad descriptions: size & shape, intensity, and texture [72],[100]. We will briefly cover the first three categories of the feature classification by Rodenacker and Bentsson categorization, as they are common to several software package implementations ([17], [58], [2]).

2.4.1 Size and shape

These features are also referred to as morphometric features, since they expresses the overall size and shape of the cell. Geometric features independent of position and orientation and thus more generally useful are area, perimeter, largest inscribe-able circle, and largest extension. There are also several shape features based on analysis of the contour of the object, such as the curvature and bending energy (measured as the rate of change in orientation of successive equidistant points on the closed curve along the perimeter) and convex hull. Also common are geometric moments that describe the extent of the object.

Another measure of shape that is used for objects are the Zernike moments [17]. In general, image moments are weighted averages of the intensities of the pixel values for an image. Zernike moments are weighted average of the intensities of objects transformed so they are within a unit circle, multiplied by the complex conjugate of the Zernike polynomial of some degree. If the image intensities are denoted by $f(x, y)$ for all valid x, y on the unit circle transformed image, then the Zernike moment Z_{nl} is defined as

$$Z_{nl} = \frac{n+1}{\pi} \sum_x \sum_y V^*_{nl}(x, y) f(x, y)$$

where $x^2 + y^2 \leq 1$, $0 \leq l \leq n$, $n - l$ is even. $V^*_{nl}(x, y)$ is the complex conjugate of a Zernike polynomial of degree n and angular dependence l,

$$V^*_{nl}(x, y) = \sum_{m=0}^{(n-l)/2} (-1)^m \frac{(n-m)!}{m! \left[\frac{(n-2m+l)}{2}\right]! \left[\frac{(n-2m-l)}{2}\right]!} (x^2 + y^2)^{(n/2)-m} e^{\frac{i}{\theta}}$$

where $0 \leq l \leq n$, $n - l$ is even, and $\theta = \tan^{-1}(\frac{y}{x}), i = \sqrt{-1}$. The Zernike polynomials form a basis over the unit circle and can capture the measure of shapes. In practice, since the moments are not invariant to rotation, the magnitude of the Zernike moments are used, calculated up to a certain order. Since shape is often an important indicator of cell type or function, some studies develop shape specific features to characterize objects. For example, Rohde et al. defined an interpolative distance over nuclei to characterize the shape variation in HeLa cell nuclei [101].

2.4.2 Intensity

Intensity measurements in the image are important for phenotype quantification. For each object, different moments of the normalized intensity distribution are calculated, and serve as features. These moments may be calculated not just for the object, but also for border regions surrounding the object, and along the contour. Sometimes different percentiles such as the value of the 10-th and 90-th are also recorded as features. Other functions of the intensity distribution includes the integrated intensity (total sum).

2.4.3 Texture

Texture features attempt to quantify the changes in intensity inside the objects. These typically take the form of various families of transformations applied to the intensity distribution of the object, in different channels or on different scales. Three commonly used transformations are the gradient, Laplace, and median. The gradient transformation of size n is an $n \times n$ filter that tries to approximate

$$|\left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right)| \tag{2.1}$$

These transformations quantify the magnitude of the changes in the intensity distribution. Laplace transformations represent an approximation to

$$\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{2.2}$$

, the sum of the second partial derivatives, and can be interpreted as the measuring the change in the gradient over the intensity distribution. There are many more transformations that are considered texture features, see [100] for more details. One often noted disadvantage of texture features is that they are not easily recognizable visually. Also, Rodenacker and Bengtsson [100] report that they may be sensitive to proper focus of the microscope.

2.5 Machine learning methods

Data from HCS experiments are images of cells expressing fluoresence in space and time. Identifying regular patterns within these images provides clues essential for understanding the underlying biology visualized by the experiments. Machine learning provides a set of tools for pattern recognition which can be applied to this data. Below, we review three families of problems in machine learning that arose during the analysis of HCS data, and describe algorithms that solve them.

2.5.1 Dimensionality reduction

Dimensionality reduction aims to transform very high dimensional data into a more compact, low dimensional representation, while limiting the loss of information contained within the data. Done right, this will strip the data of spurious variation and noise, while preserving the information necessary for further learning tasks. In many HCS studies, it is common to begin by measuring large amounts of features which have biologically interpretable meaning, but many of which have no bearing on the hypotheses the experiment seeks to test. This results in noisy measurements and makes it difficult to perform subsequent tasks like classification or clustering. One branch of solutions is to define a transformation that acts on the data, to represent it in a lower dimensional space. While mostly this results in a loss of interpretable features, it can often boost the ratio of signal to noise for a downstream learning problem [6][66]. Here we review two classic linear methods, and two more recent non-linear methods. We follow with a review of a more recent development in dimensionality reduction called representation learning, and describe autoencoders, a family of methods for representation learning.

Linear methods

Principal component analysis Principal component analysis (PCA) is a widely used technique for both dimensionality reduction and data visualization. It is most commonly defined as the orthogonal projection of the original data onto a lower dimensional subspace, subject to the constraint that the variance of the data is maximized post projection [15]. Consider a dataset $(x_1, \dots, x_N)$ and each x_i has dimension D. PCA aims to project the data onto a space having dimension $M << D$ while maximizing the variance of the projected data. In some formulations, M is a parameter that is learned during the procedure, but for now assume M is given. The variance of the dataset is the sample covariance matrix $S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \overline{x})(x_n - \overline{x})^T$. The first principal component p_1 is the vector that maximizes the projected variance $p_1^T S p_1$. The objective is a maximization problem over the vector p_1, where the objective is:

$$p_1^T S p_1 + \lambda_1(1 - p_1^T p_1)$$

$$\text{subject to } ||p_1|| < \infty$$

This can be done by differentiation with respect to p_1. The solution is seen to be that vector where

$$Sp_1 = \lambda_1 p_1$$

so p_1 is an eigenvector (with the largest eigenvalue) of S. This is called the first principal component. Each additional principal component is defined similarly, and the optimal projection onto an M dimensional subspace is a projection into the subspace with the basis of the M eigenvectors with the largest eigenvalues. PCA was used by Bakal et al. [6] to visualize their dataset under a set of selected features. They selected 12 features and projected these onto the first 3 principal components. Ultimately they abandoned the use of principal components, as the principal components in the transformed space cannot be easily interpreted biologically, in contrast to the original 12 features.

Multidimensional scaling Multidimensional scaling (MDS) is a technique that computes a low dimensional representation of a high dimensional data under the constraint that the inner products of the input data are preserved. As with PCA, the input is a dataset $(x_1, \dots, x_N)$ and each x_i has dimension

D. MDS seeks to discover a lower dimensional representation $(y_1, \ldots, y_N)$, where each y_i has dimension $M << D$ such that the sum of the Gram matrix element-wise squared difference is minimized:

$$\min \sum_{ij} (\langle x_i, x_j \rangle - \langle y_i, y_j \rangle)^2$$

The solution to this is obtained from the eigenvector decomposition of the Gram matrix from the input values, $G_{ij} = \langle x_i, x_j \rangle$. Let the top M eigenvectors of G be $\{\nu_k\}_{k=1}^{M}$ and the corresponding eigenvalues be $\{\lambda_k\}_{k=1}^{M}$. Then the output of the MDS algorithm is a dataset where

$$y_i = (y_{i1}, \ldots, y_{iM}) \tag{2.3}$$

$$y_{ik} = \sqrt{\lambda_k} \nu_{ki} \tag{2.4}$$

Non-linear methods

Also known as manifold learning methods or graph-based methods, these more recent developments formulate the dimensionality reduction problem as a graph based matrix decomposition rather than a data covariance matrix decomposition [107]. They are related under the common assumption that the initial data is a high dimensional representation that has been up-sampled from a low dimensional manifold. They each begin by constructing a graph based on the data where the nodes are data points, and edges represent either distances or similarities between data points. These graphs are intended to approximate the manifold that is sampled (in a high dimensional space) by the dataset. Two of these methods are Isomap [31], and Local Linear Embedding [103]. The notation for the high dimensional data points and low dimensional outputs remains the same as in 2.5.1. Another method that looks promising is Maximum Variance Unfolding [130], though it is not reviewed here.

Isomap Isomap is based on computing a low dimensional representation of a high dimensional dataset that most faithfully preserves the pairwise distances between input patterns, but using the distances between points as measured in the graph representation of the manifold, rather than the euclidean distances between the points in the dataset. The algorithm is simple to state, and has three steps:

1. Compute the k-nearest neighbours for each point in the dataset, and construct a graph where the nodes are the points, and the edges connect k-nearest neighbours. The edges are weighted by the distances between the neighbours.

2. Compute the pairwise distances between all the nodes in the graph, $\{D_{ij}\}$. This can be done by the all shortest paths algorithm.

3. Run MDS on the input nodes, but using the $\{D_{ij}\}$ from the previous step instead of the inner products.

Local linear embedding Local Linear Embedding (LLE) is based on computing a low dimensional representation of a high dimensional dataset that preserves the local linear structure of nearby data points. The algorithm differs significantly from Isomap and maximum variance unfolding. Instead of using the largest eigenvectors in the Gram matrix, LLE uses the lowest eigenvectors in a different matrix. The algorithm has three steps:

1. Compute the k-nearest neighbors of each point in the dataset x_i. The graph here is a directed graph whose edges indicate nearest neighbour relations.

2. Compute a (sparse) matrix of weights W_{ij} for the edges on the graph. The idea here is that LLE seeks to find a way to represent each point x_i as a linear combination of its k-nearest neighbours. The weights are chosen to minimize the reconstruction or representation error

$$\sum_i \|x_i - \sum_j W_{ij} x_j\|^2 \tag{2.5}$$

subject to the conditions that $W_{ij} = 0$ if j is not one of the k-nearest neighbours of k, and $\sum_j W_{ij} = 1$. The matrix W is a sparse encoding of the geometric distance between each data point x_i and its k-nearest neighbours.

3. Compute $(y_1, \ldots, y_N)$ where $y_i \in \mathbb{R}^M$ such that the structure of the k-nearest neighbours graph of the y_i matches W as closely as possible. This is performed by minimizing the cost function

$$\sum_i \|y_i - \sum_j W_{ij} y_j\|^2 \tag{2.6}$$

subject to the constraints that the outputs are centred $\sum_i y_i = 0$ and they have a unit covariance matrix $\frac{1}{N} \sum_i y_i y_i^T = I_M$. This minimization can be performed in several ways, but in [106] Roweis and Saul show that the solutions correspond to the largest M of the bottom $M + 1$ eigenvectors of the matrix $(I - W)^T (I - W)$ where I is the NxN identity. This is because the smallest eigenvector has all equal components, and must be discarded to satisfy $\sum_i y_i = 0$.

Representation learning

Consider a standard problem instance in machine learning, where we draw a sample dataset (input,target) tuples $D = (x_1, t_1) ..., (x_n, t_n)$. D is assumed to be a composed of i.i.d. samples from some unknown distribution $q(X, T)$ with has corresponding marginals $q(X)$ and $q(T)$. In keeping with the notation of Larochelle, Vincent and Bengio [125], we let $q_0(X, T)$ and $q_0(X)$ represent the empirical distributions of the samples in D. X is a d-dimensional random vector over $\Re^d$ or $[0, 1]^d$. T may be either discrete or real, and in addition may be unobserved. The task of *representation learning* is to find a transformation of X into Y that has more desirable properties. Analogously, Y is a d'-dimensional random vector over $\Re^{d'}$ or $[0, 1]^{d'}$. Usually, we will prefer that $d > d'$, making this an under-complete representation. However, it may also be advantageous to require $d < d'$, which is called an over-complete representation. In representation learning, Y is linked to X by some mapping $q(Y|X; \theta)$, which may be either deterministic or probabilistic, and is paramareterized by θ. How do we choose a mapping $q(Y|X; \theta)$, and learn its parameters? One principle that drives the choice of representation is that it should preserve as much information about X as possible, which can be formalized as maximizing the mutual information of X and Y $I(X; Y)$. Decomposing the mutual information into the entropy of X and the conditional entropy of $Y|X$, we have $I(X; Y) = H(X) - H(X|Y)$, which leads to the formulation:

$$\arg\max_\theta I(Y; X) = \arg\max_\theta \; -H(X|Y) \tag{2.7}$$

$$= \arg\max_\theta \mathbb{E}_{q(X,Y)}[\log(q(X|Y))] \tag{2.8}$$

where the first step follows since $H(X)$ is not affected by theta and can be treated as a constant. While q is unknown, it follows from the definition of the KL divergence that for any distribution $p(X|Y)$ we have:

$$\mathbb{E}_{q(X,Y)}[\log(p(X|Y))] \leq \mathbb{E}_{q(X,Y)}[\log(q(X|Y))] \tag{2.9}$$

This allows us to focus on picking an approximating distribution $p(X|Y,\theta')$, and then solving the following optimization:

$$\arg\max_{\theta,\theta'} \mathbb{E}_{q(X,Y;\theta)}[\log(p(X|Y;\theta'))]$$

which when combined with equation 2.9 amounts to maximizing a lower-bound on the $-H(X|Y)$, and thus on the mutual information between X and Y. How $p(X|Y;\theta')$ is parameterized will determine the form of the optimization. Below, we introduce the family of models known as autoencoders, where the transformation of X into Y is determined by a parameterized function.

Auto-encoders Auto-encoders are a family of methods that explicitly define a parameterized mapping function to encode and subsequently decode the original data X. The encoded data Y is computed as a function of X, parameterized by θ: $Y = f_\theta(X)$. The choice of this deterministic function corresponds to assuming the form of $q(Y|X;\theta)$ is a point mass $q(Y|X;\theta) = \delta(Y f_\theta(X))$. With the restriction on the form of q, the optimization over the parameterized functions can be written as

$$\arg\max_{\theta,\theta'} \mathbb{E}_{q(X)}[\log p(X|Y = f_\theta(X);\theta')]$$

which again draws on the justification of maximizing a lower bound on the mutual information. To actually evaluate candidate solutions, we replace the unknown true $q(X)$ with a sampling distribution from D

$$\arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X)}[\log p(X|Y = f_\theta(X);\theta')] \tag{2.10}$$

Practically, autoencoders are composed of both an encoder and a decoder. The encoder is a deterministic function $f_\theta(x)$ which transforms an input vector x into hidden representation y. Traditionally this takes the form of an affine mapping followed by an element-wise function s:

$$f_\theta(x) = s(Wx + b)$$

$\theta = W, b$, where W is a $d' \times d$ matrix of weights and b is a d' dimensional bias vector. The decoder is the function which maps the d' dimensional encoded data y back to the original d dimensional space.

$$z = g_{\theta'}(y)$$

. The decoding function g is also an affine transformation optionally composed with an elementwise function

$$g_{\theta'}(y) = s(W'y + b')$$

with $\theta' = W', b'$ defined as a $d \times d'$ matrix and a d dimensional vector respectively. Though the encoder and decoder are deterministic functions, the value z is thought to be a draw from a distribution $p(X|Z = z)$; if θ, θ' maximize equation 2.10, then p will generate x with high probability. Both encoder and decoder are connected by different conditional distributions of X, as $p(X|Y = y) = p(X|Z = g_{\theta'}(y))$.

This leads finally to the idea of reconstruction error, which is a loss function that may be optimized:

$$L(x,z) \propto \log p(x|z) \tag{2.11}$$

The domain of X influences which choices for $p(x|z)$ and $L(x,z)$ are most appropriate.

- For $x \in \Re^d$, $p(x|z)$ usually distributed $N(z, \sigma^2 I)$. This suggests the familiar squared error loss $L(x,z) = L_2(x,z) = C \parallel xz \parallel^2$, where C is a constant with σ^2 folded in, since it will not affect the choices of θ, θ'.
- For $x \in [0,1]^d$, $p(x|z)$ is distributed multivariate Bernoulli $B(z)$. This suggests the cross-entropy loss $L(x,z) = -\sum_j [x_j \log(z_j) + (1x_j)\log(1z_j)]$.

Training an autoencoder is carried out by minimizing the chosen reconstruction error. Autoencoder training consists of minimizing the reconstruction error with respect to the parameters:

$$\arg\min_{\theta,\theta'} \mathbf{E}_{q^0(X)}[L(X,Z)] \tag{2.12}$$

$$\arg\min_{\theta,\theta'} \mathbf{E}_{q^0(X)}[\log p(X|Z = g_{\theta'}(f_\theta(X)))] \tag{2.13}$$

This is a rather long way of saying that minimizing the reconstruction error of an autoencoder with deterministic encoder and decoder functions amounts to maximizing a lower bound on the mutual information between X and Y. Autoencoders share connections even with more complex probabilistic models such as Restricted Boltzmann Machines (RBM), even though they are parameterized by deterministic functions. If an autoencoder uses a sigmoid function to apply elementwise in both the encoder and decoder, it shares the same form as the conditional distributions $P(h|v), P(v|h)$ of binary RBMs [10]. The training procedures differ greatly, with RBMs maximizing a likelihood directly (or approximating a joint distribution through contrastive divergence). This reveals an immediate advantage for autoencoders: minimizing a reconstruction function is both interpretable and stable to optimize.

De-noising autoencoders The autoencoder model described above is the simplest form of the model, which was originally proposed for dimensionality reduction. This meant the dimensionality of the encoded data d' was chosen to be much smaller than that of the original data d. However, other representation models such as sparse coding and RBMs have achieved success when using an over-complete configuration where $d' > d$. For an autoencoder, over-complete representations risk encouraging optimizers to favour a trivial solution where the identity map is the result. While this achieves a global minimum in the reconstruction objective, it defeats the purpose of learning a new representation. To allow for intentionally over-complete representations to be learned, additional constraints have been developed to further regularize the learning of the encoding and decoding parameters.
One such additional constraint introduced by Vincent et al. has been the deliberate injection of noise into X prior to the encoding process [124, 125]. Instead of receiving the input X, a *denoising* autoencoder (dA) gets input $\tilde{X}$, which is a partially corrupted version of the input. This is accomplished by drawing from a distribution $\tilde{\mathbf{x}} \sim q(\tilde{X}|X)$.

Learning in a dA model involves minimizing noisy reconstruction error

$$\arg\min_{\theta,\theta'} E_{qX,\tilde{X}} \left[L\left(X, g_{\theta'}(f_\theta(\mathbf{x}_i))\right)\right] \tag{2.14}$$

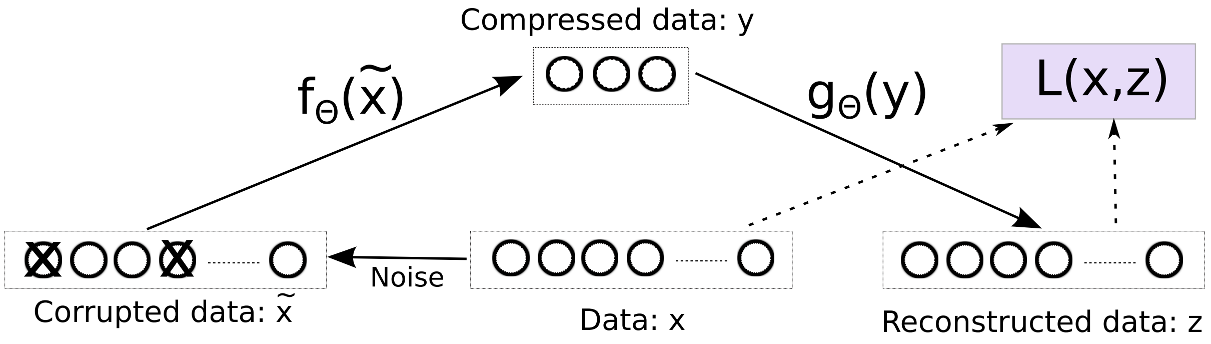


Figure 2.4: A diagram of how the denoising autoencoder goes from input data, to masked data, to latent representation, and finally to reconstructed input.

where the expectation is over the joint distribution of inputs and corrupted inputs. The optimization must find a solution where the reconstructed data is close to the corrupted data, but also closer to the original un-corrupted sample. An appealing justification for this choice of regularization is derived from a geometric interpretation of the data. The manifold assumption [22] states that natural high dimensional data concentrates close to a non-linear low-dimensional manifold. The injection of noise into the high dimensional data given to the autoencoder forces it to learn a model that maps corrupted data close to the manifold.

2.5.2 Supervised learning

Supervised learning is the machine learning task of learning a function from a set of data to a defined set of outputs or labels, where the learning is guided by a set of training data. Let $\mathbf{X} = (x_1, \ldots, x_n)$ be a set of n examples that constitute the training data, where each $x_i \in \Omega$. Each x_i has a corresponding y_i, where $y_i \in \Upsilon$ are called the labels of the examples x_i. Usually it is assumed that the training data (x_i, y_i) are sampled i.i.d (independently and identically distributed) from a distribution defined over $\Omega \times \Upsilon$. When $\Upsilon = \mathbb{R}$ or $\Upsilon = \mathbb{R}^d$, or more generally for any continuous space, then the task is called regression. If the y_i are drawn from a finite discrete set, then the task is called classification, and they are referred to as class labels. Another distinction among supervised learning algorithms are generative algorithms versus discriminative algorithms. *Generative* algorithms attempt to model the probability distribution that generates the observed data, often given the class label $p(x|y)$. Then, using Bayes rule, the probability that a given x is associated with any y is given by

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy} \tag{2.15}$$

The numerator in (2.15) also represents the joint density of the data $p(x, y)$, from which you can imagine (x_i, y_i) pairs being generated. *Discriminative* algorithms do not try to estimate $p(y|x)$ by modeling the joint distribution of the data. Instead they focus on other ways to correctly predict y for a given x, often times just predicting the most likely y. I will review one popular example of each type of model here, and point out some examples where they have been applied.

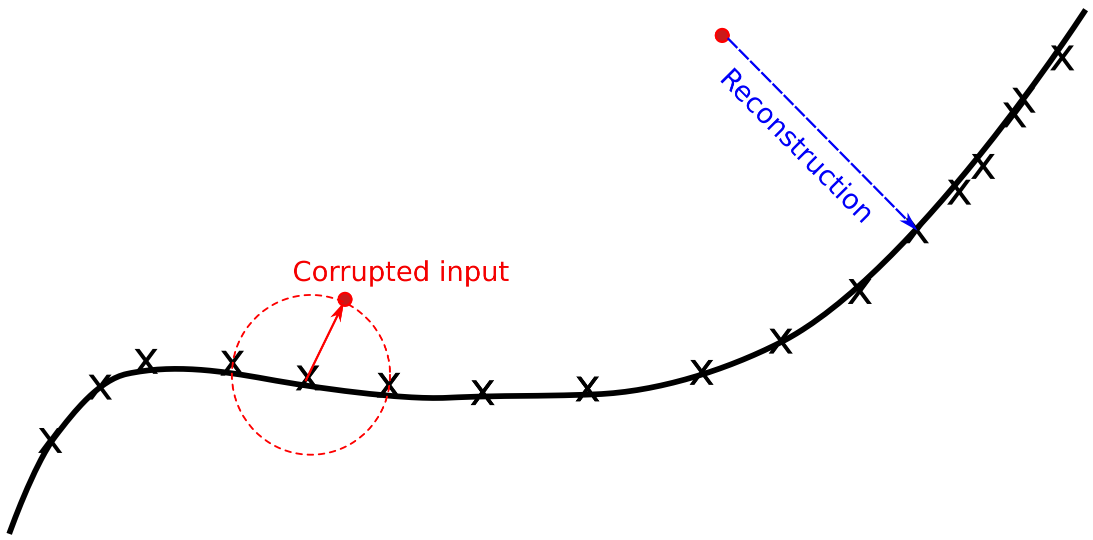


Figure 2.5: Sample diagram of data which exists on or near a low-dimensional manifold. Points sampled from the generating distribution $q(X)$, denoted by X are selected at random during training. Noise is added, taking the points away from the manifold (red). The denoising autoencoder must learn to take the noisy data and reproduce a sample must close to the manifold (blue).

Support vector machines

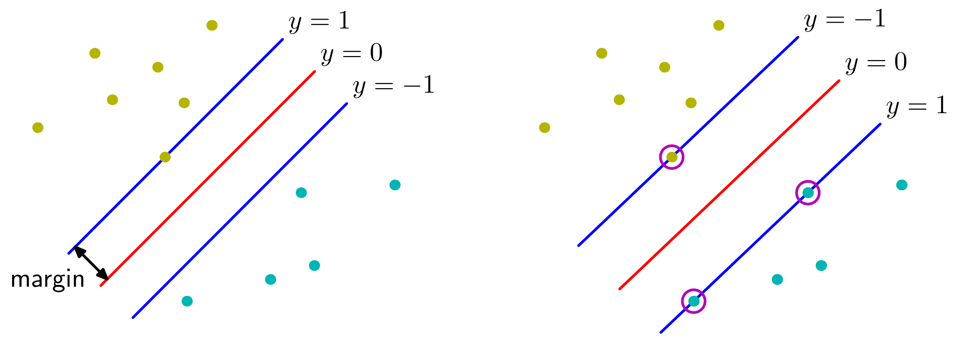
Discriminative algorithms focus on achieving the most effective representation of $p(y|x)$ without modeling the joint distribution of the data. As compared to generative models, there are usually fewer parameters to fit, and therefore the learning task may be easier. Predictive performance may also improve, especially if there the estimate of class conditional densities, the $p(C_i|x)$ upon which generative models are built, are not good estimates of the true distributions [15].

For simplicity, let us consider only a two-class classification problem. Denote the dataset by $\mathbf{X} = (x_1, \ldots, x_n) \in \mathbb{R}^M$, with corresponding class labels $y_i \in -1, +1$. Now imagine the points in the the dataset dispersed around the M dimensional space. The support vector machine classifier (SVM) is a discriminative algorithm that finds a hyperplane which best separates the two classes. A hyperplane in $\mathbb{R}^M$ consists of the set of points x that satisfy the linear equation:

$$f(x) = \beta^T x + \beta_0 = 0$$

The hyperplane is called a separating hyperplane there exists $c > 0$ such that

$$y_i(\beta^T x_i + \beta_0) \geq c \,\forall\, i = 1, \ldots, n$$



(a) Separating hyperplane, ok **(b)** Separating hyperplane, best

Figure 2.6: Reproduced from Bishop [15], this shows a dataset that is separated by two hyperplanes (shown in red). In 2.6a, the hyperplane separates the two classes, but the margin is not maximal. The hyperplane in 2.6b is maximal, and shows the points which are support vectors as circles.

For convenience, usually c is chosen as 1. If the classes can be separated by a hyperplane, then the best way to maximize the performance of the hyperplane as a classifier is to construct it so that it lies as far from the training data as possible. This intuition is formalized as the *margin* of the hyperplane

$$\text{margin} = 2 \times \min \{y_i d_i, i = 1, 2, ..., n\}$$

where d_i is the signed distance of the point x_i from the hyperplane.

The optimal separating hyperplane (with maximal margin) is found by minimizing:

$$\frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{n} \xi_i \tag{2.16}$$

$$\text{subject to: } y_i(\beta^T x_i + \beta_0) \geq 1 - \xi_i \ \ , \ \xi_i > 0 \ \forall i \tag{2.17}$$

The variables ξ_i are introduced to relax the original constraint of a separating hyperplane, because it is not always the case that the two classes are completely separable. The term $\gamma \sum_i \xi_i$ controls the degree of relaxation, with γ acting as a tuning parameter.

The real power of SVMs comes from what is referred to as the *kernel trick*. The solution to equation 2.16 can be re-parametrized so that the optimal plane is given by

$$\beta = \sum_{i \in SV} \alpha_i y_i x_i$$

where SV is the set of data points for which the new parameters α_i are strictly positive. These coefficients are the solution to a dual optimization problem

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{2.18}$$

$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0 \ \forall i \tag{2.19}$$

The solution to optimization 2.18,2.19 depends on the data only through the inner products $x_i^T x_j$. Instead of the inner product, another definition of similarity $K(x_i, x_j)$ can used that maps the data to a different feature space. Crucially, classes which were not linearly separable in the original space may become so in the range of K. This replacement of the inner product with a representative from a family of *kernel* functions is the kernel trick. The choice of K is arbitrary, so long as it conforms to the definition of a kernel function (See [134]), and is non-negative definite.

Support vector machines have been used effectively for classification problems in high content imaging studies. Fuchs et al. [38] used SVMs as part of an analysis pipeline to predict gene function from the morphology of HeLa cells. They used the one versus one method for multi-class SVM classification (with majority voting) to classify cells into one of 10 predefined morphological classes. SVMs were also used by Neumann et al. [86] in their study of cell division genes. They performed time-lapse microscopy profiling of live HeLa cells where histone 2B was marked with mCherry (a red fluorescent protein) and tubulin (a cytoskeletal protein) was marked with GFP, and subjected them to approximately 21000 different gene knockdown conditions by siRNA. SVMs were used to classify cells into one of 16 pre-defined morphological classes.

2.5.3 Unsupervised learning

Unsupervised learning is the machine learning task where no label information is associated with the input data. In contrast to supervised learning, unsupervised learning algorithms must take the unlabeled data as input, and discover some kind of interesting structure. Let $\mathbf{X} = (x_1, \ldots, x_n)$ be a set of n data points of dimension d drawn i.i.d from some distribution defined over Ω. Unsupervised learning algorithms make different assumptions about how this data may have been generated, and then search for a distribution over Ω that could likely have generated $\mathbf{X}$. In this section, I will review two different models for unsupervised learning. The first is k-means clustering, a well studied and used method for clustering. Next I will review mixture of gaussian clustering, which extends k-means into a more flexible clustering framework.

K-means

K-means clustering seeks to partition a dataset $\mathbf{X} = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}^d$ into K disjoint subsets called clusters. These clusters represent dense pockets where the data points within a cluster are closer to each other than to points outside the cluster. Each cluster is represented by a 'centre' $\mu_k \in \mathbb{R}^d$. The K-means clustering algorithm seeks to find an assignment of data points to clusters such that the sum of the squared distances from each data point to its cluster centre is minimized. Indicator random variables r_{nk} are introduced to keep track of the assignment of points to clusters: $r_{nk} \in \{0, 1\}$, with $r_{nk} = 1$ if data point x_n belongs to cluster μ_k. These are called the *responsibilities*. The goodness of a clustering assignment in K-means is measured by the *distortion*:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{K} r_{ik} \|x_n - \mu_k\|^2 \tag{2.20}$$

which measures the sum total of the squared distances from each data point to its cluster centre. The algorithm initially assigns randomly chosen values to the K centres, then iteratively performs the following two operations:

1. Update the responsibilities: minimize J with respect to r_{nk} while keeping the centres fixed
2. Update the centres: minimize J with respect to μ_k while keeping the responsibilities

These two update steps are repeated until the solution converges. It is worth noting that there is a well established correspondence between this formulation of K-means clustering and the mixture of Gaussians model in section 2.5.3. If the K Gaussian random variables are all defined with parameters $(\mu_k, \Sigma = \sigma \times I)$, where σ is a shared variance parameter and I is the identity matrix, then as $\sigma \to 0$ the posterior probabilities $\gamma(z_k)$ in the mixture of Gaussians model focus all density on one particular Gaussian mixture component, and in the limit they approach the same distribution as the indicator r_{nk} variables. See [15] for full details.

K-means has other variants. For example, Levy and Siegal [66] used a variant called K-medoids (also known as the Partitioning around Medoids or PAM algorithm [59]) in their study of phenotypic variation in yeast. They performed gene knockout experiments on yeast, then imaged and classified each cell into one of 220 phenotypes. They they used the K-medoids algorithm with differing values for K to reduce the number of phenotype classes to around 70.

Gaussian mixture models

Generative models attempt to model the distribution from which the data X may have plausibly been sampled. The mixture of Gaussians model is a generative model where the data is assumed to have been generated by repeated sampling from a number of different Gaussian random variables. Denote the dataset by $\mathbf{X} = (x_1, \ldots, x_n)$. In a Gaussian mixture each data point is generated by a linear mixture of different Gaussian random variables

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{2.21}$$

An important notion in this formulation is that each data point is assumed to have been sampled from exactly one of these K Gaussian random variables, but that this information is unobserved. This binary vector latent random variable is denoted $\mathbf{z}$, where if $z_k = 1$ then all the remaining elements are zero, which means that $\sum_k z_k = 1$ and so the $\mathbf{z}$ distribution is well defined, with marginal distribution $p(z_k = 1) = \pi_k$, or the proportion of variables from the dataset that are sampled from the kth Gaussian. The π_k parameters must therefore satisfy the requirements for a probability distribution, namely $0 \leq \pi_k \leq 1, \sum_{k=1}^{K} \pi_k = 1$. Since the marginal distribution over $\mathbf{z}$ can also be represented as a product $\prod_{k=1}^{K} \pi_k^{z_k}$ (which works since the z_k are binary), this means the conditional distribution of a data point given a particular $\mathbf{z}$ can be written in the form

$$p(x|\mathbf{z}) = \prod_{k=1}^{K} N(x|\mu_k, \Sigma_k)^{z_k}$$

The joint distribution of the data is given by $p(x, \mathbf{z}) = p(x|\mathbf{z})p(\mathbf{z})$, and so the marginal distribution of the data can be expressed as

$$p(x) = \sum_{z} p(x|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$$

What makes mixture of Gaussians so useful as a generative model for classification is that once the model is trained, and new unlabeled data x_{n+1} is to be classified, Bayes rule can be used to determine the conditional probability of the class label, which in this case is the Gaussian mixture component most likely to have generated x_{n+1}:

$$\gamma(z_k) \equiv p(z_k = 1|x_{n+1}) = \frac{p(z_k = 1)p(x_{n+1}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x_{n+1}|z_j = 1)} \tag{2.22}$$

$$= \frac{\pi_k N(x_{n+1}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_{n+1}|\mu_j, \Sigma_j} \tag{2.23}$$

The variables $\gamma(z_k)$ for $k = 1...K$ are often called the responsibilities, as they represent the posterior probability distribution over Gaussian mixture components responsible for generating x_{n+1}. To classify x_{n+1}, a decision must be made using the responsibilities. One common choice is to select the maximum over the posterior of the responsibilities for each data point, which is called the maximum a posteriori estimate.

In the learning phase, mixture of Gaussian models are given the mixing parameters π_k but must learn the optimal specification of all the Gaussians $(\mu_1, \ldots, \mu_K)$, $(\Sigma_1, \ldots, \Sigma_K)$. This is often done by maximizing the likelihood of the data $p(\mathbf{X}|\pi, \mu, \Sigma) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \Sigma_k)$. The preferred method for maximizing the likelihood is by the expectation maximization (EM) method [32][83]. This is a general purpose method for maximizing the likelihood $p(\mathbf{X}|\theta)$ of a joint distribution $p(\mathbf{X}, \mathbf{Z}|\theta)$ in terms of the parameters θ.

Mixture of Gaussian models fit quite naturally in classification problems for HCS. In their study of yeast spindle morphogenesis, Vizeacoumar et al. [126] used mixture of Gaussians modeling to classify morphological phenotypes describing spindle pole body formation into 60 different classes. They identified strains containing mutations that induced significantly different spindle pole phenotypes compared to wild type. Another example is a study by Ranzato et al. [95], that presents a general method for recognizing and classifying biological particles. Their work involves using a mixture of Gaussians model for classifying biological objects that have been transformed into a vector of features. They fit their model using EM to estimate the parameters, and a uniform prior distribution for the π_k.

Nonparametric mixture models

Previous k-means and Gaussian mixture models share one key assumption, which is that the number of components in the mixture is known apriori. This quantity, k, becomes a hyper-parameter for both algorithms, and must be estimated in one of several different ways. One way is simply use cross-validation to fit the model parameters on folds of the data, using different values for k, and average the resulting loss obtained over all the folds fit to a given value for k as a measure of goodness. For k-means, this procedure would set k, and then randomly initialize the model several times, run EM until convergence, and averaging the distortion values that result. For Gaussian mixtures, it would repeat the same procedure, but would use the likelihood of final parameter values. This cross-validation procedure may be combined with a regularization scheme to guard against over-fitting by penalizing models with many components, such as the AIC or BIC. Bayesian non-parametrics (BNP) provide an alternative. Rather than fit many models with different amounts of complexity, the Bayesian non-parametric approach is to fit one model which adapts its complexity to suit the data. For example, when using a BNP approach for mixture modeling, a Bayesian model estimates the number of components needed to best explain the dataset. All

Bayesian non-parametric approaches are at heart generative models with hidden variables. Each BNP model specifies the relationship of dependencies between all the variables, both hidden and observed. It also specifies the process by which the observed data is generated, which includes prior distributions over model parameters and hidden variables. Analyzing data in a BNP takes the form of computing the *posterior distribution* of the model variables conditioned on the data [40]. The BNP alternative to parametric mixture modeling is called the Dirichlet process mixture model.

Dirichlet process mixture model

The Dirichlet process mixture model (DPMM) [4] is one of the most common BNP models. It is built on the Dirichlet process, which is a stochastic process that is a distribution over distributions: each random draw from a Dirichlet process yields a finite distribution over some probability space Θ. Let H be any distribution over Θ, and $\alpha \in \Re$ be a positive real value. Then consider any finite measurable partition of Θ, $A_1, \dots, A_r$. For any distribution G, we say that G is distributed according to a Dirichlet process if the vector $(G(A_1), \dots, G(A_r))$ is distributed according to a specific Dirichlet distribution:

$$(G(A_1), \dots, G(A_r)) \sim Dir(\alpha H(A_1), \dots, \alpha H(A_r)) \tag{2.24}$$

if this is true for every finite measurable partition of Θ, then we say G is Dirichlet process distributed, with base distribution H and concentration parameter α: $G \sim DP(H, \alpha)$ [117]. H can be interpreted as the mean of the DP, since $\mathrm{E}[G(A)] = H(A)$ for any set A. The parameter α acts as an inverse variance: $Var(G(A)) = H(A)(1 - H(A)/(\alpha + 1)$. Essentially, the larger α becomes, the smaller the variance, and the more the DP will concentrate about its mean. Upon first reading, the Dirichlet process can seem a bit abstract. One way to make it more concrete is to show how a simpler model, the Bayesian finite mixture model, extends a model we have already seen: mixture of Gaussians. We then go on to show that the DPMM extends Bayesian finite mixtures.

Bayesian finite mixtures

In a Bayesian finite mixture, there are K components, each with parameters θ_k. Each element of the dataset, X_i, is generated by first selecting a component c_i according to some distribution over components $P(c_i)$, and then sampling X_i from the emission distribution for component c_i, $F(\theta_{c_i})$. The emission distributions will differ depending on the underlying observed data being modeled; for real valued data, Gaussian distributions are a natural fit. For Bayesian mixture models, there are additional priors over the mixing distribution $P(C)$, as well as over the emission distribution parameters $\theta \sim G_0$. With priors chosen for both these sets of parameters, the joint distribution over observed data $\mathbf{X} = (X_1, \dots, X_N)$, cluster assignments $\mathbf{c} = (c_1, \dots, c_N)$ and emission parameters θ can be specified:

$$P(\mathbf{X}, \mathbf{c}, \theta) = \prod_{k=1}^{K} G_0(\theta_k) \prod_{i=1}^{N} F(x_i|\theta_{c_i}) P(c_i) \tag{2.25}$$

which presumes that the distribution over each data point is independent of all the others, given the latent cluster assignment. It is these assignments of data points into clusters that we try to infer by posterior inference. The posterior $P(\mathbf{c}|\mathbf{X})$ is calculated using Bayes' rule:

$$P(\mathbf{c}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{c})P(\mathbf{c})}{\sum_{\mathbf{c}} P(\mathbf{X}|\mathbf{c})P(\mathbf{c})} \tag{2.26}$$

the numerator in this fraction consists of the likelihood of the data multiplied by the prior over cluster assignments. The likelihood can be obtained by marginalizing over settings of the parameters θ:

$$P(\mathbf{X}|\mathbf{c}) = \int_\theta \left[\prod_{i=1}^{N} F(\mathbf{X}|\theta_{c_i}) \prod_{k=1}^{K} G_0(\theta_k) \right] d\theta \tag{2.27}$$

which can be computed analytically if G_0 is chosen to be conjugate to the parameters of F. If we examine the numerator of 2.26, we can see the connection of the Bayesian finite mixture with the simpler mixture of Gaussians model: the likelihood term first seen in section 2.5.3 appears in a similar calculation used to calculate the responsibilities of each component for the data points. Where as more care is taken to specify the prior distributions over shape in Bayesian finite mixtures, mixture of Gaussians cares only about which settings of the parameters for each of the components maximizes the likelihood.
The posterior of assignments given the data (equation 2.26) cannot be explicitly computed, as this would require summing over all possible partitions of the data into K groups that appears in the denominator (this term is called the marginal likelihood). This requires us to use some approximate methods, such as Markov Chain Monte Carlo sampling [85], or variational inference [16].

Extension to infinite number of mixture components

Suppose that we don't know how many components we want to include in our mixture model. The most straight-forward way is to choose different values for K, and then try to approximate the marginal likelihood of each of these models. However, a reminder of how difficult it is to even approximate the marginal likelihood means this approach is risky: computing the approximations is very expensive, and only adds to our uncertainly over which models are more appropriate. An appealing alternative is to extend the Bayesian finite mixture so the model allows for a potentially infinite number of mixture components, but which strongly favours a small number of components responsible for the data. This allows us to select the most appropriate number of components to use, as a result of doing inference on the posterior, which we can do. Let us restate the Bayesian finite mixture, as a set of random variables and their dependencies for generating each data point:

$$\begin{aligned} \pi &\sim Dir(\frac{\alpha}{K}, \dots, \frac{\alpha}{K}) \\ \theta_k &\sim G_0 \\ c_i|\pi &\sim Discrete(\pi_1, \dots, \pi_k) \\ x_i|c_i, \theta_k &\sim F(\theta_{c_i}) \end{aligned} \tag{2.28}$$

This depiction explicitly shows the prior distributions over the cluster indicators $\mathbf{c}$ and component parameters θ_k. It also shows the symmetric Dirichlet prior over mixing proportions π, each of whose parameters would go to zero as K gets large. However, as shown in [85], we can integrate over mixing proportions to get a different representation of the conditional distribution of cluster assignments:

$$\begin{aligned}
&P(c_i = c|c_1, \ldots, c_{i-1}) \\
&\quad = P(c_1, \ldots, c_{i-1}, c_i = c)/P(c_1, \ldots, c_{i-1}) \\
&\quad = \frac{\int_\pi \pi_{c_1} \ldots \pi_{c_{i-1}} \pi_c ; \Gamma(\alpha)\Gamma(\alpha/K)^{-K} \pi_1^{\alpha/K-1} \ldots \pi_k^{\alpha/K-1} d\pi}{\int_\pi \pi_{c_1} \ldots \pi_{c_{i-1}} ; \Gamma(\alpha)\Gamma(\alpha/K)^{-K} \pi_1^{\alpha/K-1} \ldots \pi_k^{\alpha/K-1} d\pi} \\
&\quad = \frac{n_{i,c} + \alpha/K}{i - 1 + \alpha}
\end{aligned} \tag{2.29}$$

where $n_{i,c}$ represents the number of data points assigned to component c (or equivalently the size of cluster c). If we now allow K in equation (2.28) to go to infinity, we have a sensible representation for the cluster assignments over the ith data point, given the assignments of points $1, \ldots, i-1$:

$$\begin{aligned}
P(c_i = c|c_1, \ldots, c_{i-1}) &= \frac{n_{i,c}}{i - 1 + \alpha} \\
P(c_i \neq c_j for j < i|c_1, \ldots, c_{i-1}) &= \frac{\alpha}{i - 1 + \alpha}
\end{aligned} \tag{2.30}$$

This conditional distribution encapsulates the ability of the DPMM to grow in capacity. The assignment of the ith data point can either be to one of the previous clusters, in proportion to their current size, or to a new cluster, in proportion to the concentration parameter α. The process of distributing a potentially infinite set of indicators to points this way is also known as the Chinese Restaurant process. This allows us to extend the generative process in equation (2.28) into one for the DPMM:

$$\begin{aligned}
G &\sim DP(\alpha, H) \\
c_i|c_{j<i} &\sim G \quad i \in 1, \ldots, N \\
x_i|c_i &\sim F(\theta_{c_i}, G_0)
\end{aligned} \tag{2.31}$$

Here, the mixing distribution G is distributed according to a DP, and the component parameters for the emission distributions $F(\cdot)$ are determined both by the prior G_0 as well as the cluster assignment. In later chapters, we will describe how to set up posterior inference for the DPMM.

2.5.4 Algorithms for inference in Bayesian nonparametric models

As we saw in section 2.5.3, exact estimation of posterior distributions for DPMM parameters is intractable, due to the exponential explosion that occurs in calculating the marginal likelihood. There are two main ways in which this problem is circumvented in order to estimate the posterior distribution. The first is by sampling from the posterior distribution of the parameters of the component distributions and component indicators. This is performed in practice by Markov chain Monte Carlo sampling methods, or MCMC. The second is to impose additional assumptions about the conditional independence of the posterior distribution which allows tractable inference, and to then minimize the divergence between this approximate posterior and the true posterior. This reduces the distribution estimation problem into an optimization problem. This family of methods is called *variational inference* [57]. Both methods have their roots in statistical physics, and both methods have advantages and disadvantages.

MCMC methods, assuming they satisfy certain conditions[2], will provide samples from the exact

[2]namely, the Markov chain they simulate must be ergodic, and each transition must leave the equilibrium distribution

posterior. The most elementary methods are also simple to describe and implement, as we will see. However, they suffer from several deficiencies. The MCMC samples are not independent, so we must be careful in how we employ them to estimate moments of the random variable in the posterior. A common choice is to thin out the samples, using only each t-th sample for some positive integer t. Another problem is that we must start the sampling process from some random guessed initial values, which influences the early exploration of the space, that biases our samples. This can be mitigated by employing a 'burn-in' period; for a threshold b, throw away the first b samples, and only consider those subsequent to b. Mitigation through sampling adds up, meaning that MCMC requires very many samples to achieve the theoretical guarantees unbiased estimates from the exact distribution. Determining when convergence has been achieved can be difficult. Variational inference, on the other hand, computes point estimates for its solutions very quickly, since it directly optimizes an objective minimizing the discrepancy between the true posterior and an approximate one. However, it is not as generally applicable as MCMC methods, and its usefulness is largely limited to exponential family models [128, 16].

Collapsed Gibbs sampling

Gibbs sampling is a simple Markov Chain Monte Carlo algorithm that operates by sampling from the conditional distributions over each of the variables in the posterior distribution [39]. Consider a distribution $p(\mathbf{z}) = p(z_1, \ldots, z_m)$ from which we will draw samples. Gibbs sampling updates the value of each variable z_i by iteratively drawing from the distribution of z_i given all the other z_{-i} variables in the joint distribution (except for z_i):

Algorithm 1: Gibbs sampling

```
input  : initial values for $z_1, z_2, z_3$
output: set of $t$ samples for $(z_1, z_2, z_3)$
for $\tau$ in $(1, \ldots, t)$ do
    sample $z_1^{\tau+1}$ from $p(z_1|z_2^\tau, z_3^\tau)$ sample $z_2^{\tau+1}$ from $p(z_2|z_1^{\tau+1}, z_3^\tau)$ sample $z_3^{\tau+1}$ from
    $p(z_3|z_1^{\tau+1}, z_2^{\tau+1})$
end
```

Suppose that we want to model data $(X_1, \ldots, X_n)$ as Dirichlet Process mixture model, specified as follows:

$$G|G_0, \alpha_0 \sim DP(\alpha_0, G_0) \tag{2.32}$$

$$\theta_i \sim G \tag{2.33}$$

$$X_i|\theta_i \sim F(\theta_i) \tag{2.34}$$

To sample from the posterior distribution over $(\theta_1, \ldots, \theta_N|X_1, \ldots, X_n)$, we need to derive the conditional posterior distributions for each parameter θ_i, conditioned on each of the others. Gibbs sampling then reduces to cycling through each data point, drawing a new value for its component parameter from the conditional distribution for that component parameter given all the other data points and parameters:

$$\theta_i = t|\theta_{-\mathbf{i}} \sim \frac{\alpha_0 G_0(t)}{\alpha_0 + n - 1} + \frac{\sum_{j \neq i} \delta(t - \theta_j)}{\alpha_0 + n - 1} \tag{2.35}$$

invariant. See section 11.2 of [15], or [84] for more details.

That is, the conditional distribution for the observation model for data point i is a mixture of two distributions: either a new value drawn from the base measure, or from a mixture of the other component values.

$$P(\theta_i|\theta_{-\mathbf{i}}, \mathbf{X_i}) = b\alpha_0 \left[\int_\theta G_0(\theta)F(X_i|\theta)\right] H(\theta_i|X_i) + b\sum_{i\neq j} F(X_i|\theta_j)\delta(\theta_i - \theta_j) \tag{2.36}$$

$$H(\theta_i|X_i) = \frac{G_0(\theta_i)F(X_i|\theta_i)}{\int_\theta G_0(\theta_i)F(X_i|\theta_i)} \tag{2.37}$$

So H is the posterior for θ given one observation, and the prior G_0, and b is a normalizing constant. In the setting where G_0 is conjugate to $F(X_i|\theta)$, then the integrals in equation 2.36 can be computed analytically. While performing Gibbs sampling in this way will work, it is very inefficient [85]. The clustering property of the DP means many data points will be generated by Fs having the same θ value. Yet under this sampling scheme, we may only change θ for one data point at a time; changing the θ values for all observations in a cluster can only occur via a joint sequence of changes during which the cluster is broken apart for each point, and then reformed around the new θ value. So while we are guaranteed to eventually arrive at a configuration of our variables that are from the correct posterior distribution, we will have to draw too many samples to do so.

From this observation, and what we know about the clustering properties of the DP, we may intuitively guess that if we re-formulate the DP so that we sample over fewer random variables, then we may be able to sample more efficiently. In fact, this intuition is well characterized in the sampling literature, especially when using models with conjugate priors. The process, known as Rao-Blackwellization[3] in the sampling literature, involves integrating out some of the unobserved variables in the model to shrink the set of variables over which we sample. Since the process involves reducing the number of random variables over which we are sampling, it is called *collapsed* Gibbs sampling. For DPMM models, the variables which are collapsed are the mixing proportions π, as well as the parameters of the observation model θ. This leaves us with only sampling over the distribution of the component indicators z_i for each data point, as specified in (background section).

Derivation of the collapsed Gibbs sampler For Gaussian mixtures with observation distributions $F(x_i|\theta_k)$ drawn from the exponential family with conjugate prior $G_0(\theta_k|\lambda)$, integrating over the mixing proportions π as well as the component parameters $\{\theta_k\}_{k=1}^K$ leaves only the cluster assignment indicators $\{z_i\}_{i=1}^n$ as unobserved. The conditional posterior for these is $p(z_i|\mathbf{z}_{-i}, \mathbf{x}, \alpha_0, \lambda)$, which factorizes as follows:

$$\begin{aligned} p(z_i|\mathbf{z}_{-i}, \mathbf{x}, \alpha_0, \lambda) &= p(z_i|x_i, \mathbf{z}_{-i}, \mathbf{x}_{-i}, \alpha_0, \lambda) && (2.38)\\ &\propto p(z_i|\mathbf{z}_{-i}, \mathbf{x}_{-i}, \alpha_0, \lambda)p(x_i|z_i, \mathbf{z}_{-i}, \mathbf{x}_{-i}, \alpha_0, \lambda) && (2.39)\\ &= p(z_i|\mathbf{z}_{-i}, \alpha_0)p(x_i|\mathbf{x}_{z_i-i}, \lambda) && (2.40) \end{aligned}$$

This uses Bayes rule to factorize the conditional posterior of z_i into a product of its prior and likelihood, and then the application of the Bayes' ball tests to reduce the number of random variables upon we must condition. The last step yields two terms. The first is the result of marginalizing over the mixing

[3]this term is unexpectedly contentious; see section 2.2.1 of [82].

proportions π [132], which yields a form which we have seen already in chapter two:

$$p(z_i|\mathbf{z}_{-i}, \alpha_0) = \frac{n_{z_i-i} + \alpha_0/K}{n + \alpha_0 - 1} \tag{2.41}$$

the probability that the z_ith component of the mixture is responsible for $n_{z_i-i} = n_{z_i} - 1$ data points, from the Chinese Restaurant Process. The second is the results of marginalizing over the θs:

$$p(x_i|\mathbf{x}_{z_i-i}, \lambda) = \int p(x_i|\theta_k)p(\theta_k|\mathbf{x}_{z_i-i}, \lambda)d\theta_k \tag{2.42}$$

which can be computed analytically. Given the conjugacy between the prior $G_0(\theta_k, \lambda = (\nu, \eta))$ and θ_k, the second term in integrand (the posterior of θ_k given its data) on the first line is a distribution that subsumes the information provided by the $\mathbf{x}$ values

$$p(\theta_k|\mathbf{x}_{z_i-i}, \lambda) = p(\theta|\tilde{\nu}, \tilde{\eta}) \tag{2.43}$$

where $\tilde{\nu} = \nu + \sum_i s(x_i), \tilde{\eta} = \eta + n$ are the sums between the values from the prior and the sufficient statistics of the data points attributed to component z_i for the distribution F. This 'new' prior over θ_k gives us a reformulated integrand

$$p(x_i|\mathbf{x}_{z_i-i}, \lambda) = \int p(x_i|\theta_k)p(\theta|\tilde{\nu}, \tilde{\eta})d\theta_k \tag{2.44}$$

that we recognize as the marginal distribution over the $\mathbf{x_i}$, and for which we know the functional form again due to conjugacy. Expanding all the distributions into their representations in the exponential family and reducing [118]:

$$\begin{aligned} p(x_i|\mathbf{x_k}, \lambda) &= \frac{p(\theta_k)p(\mathbf{x}_k|\theta_k)}{p(\theta_k|\mathbf{x}_k} && (2.45)\\ &= \frac{exp\left(t(\theta)^T\nu - \eta\psi(\theta) - \xi(\nu,\eta)\right) exp\left(t(\theta)Ts(x) - \phi(x) - \psi(\theta)\right)}{exp\left(t(\theta_k)^T(\nu + \sum_{j\in\mathbf{x_k}} s(x_j)) - (\nu + n_k)\psi(\theta_k) - \xi(\nu + \sum_{j\in\mathbf{x_k}} s(x_j), \nu + n_k)\right)} && (2.46)\\ &= exp\left(\xi(\tilde{\nu} + s(x_i, \tilde{\eta} + 1) - \xi(\tilde{\nu}, \tilde{\eta} - \phi(x_i))\right) && (2.47)\\ &\equiv f_k(x_i; S_k, n_k) && (2.48) \end{aligned}$$

we arrive finally at the form of the conditional posterior 2.38, combining the results of 2.45 and 2.41:

$$p(z_i = k|\mathbf{z}_{-i}, \mathbf{x}, \alpha_0, \lambda) = \frac{n_{k-i} + \alpha_0/K}{n + \alpha_0 - 1} \times f_k(x_i; S_k, n_k) \tag{2.49}$$

Of course in the DPMM, $K \to \infty$, so we maintain only those active components which are responsible for at least one data point. The sufficient statistics for our Gaussian components (i.e the observation model) are the sum and squared sum of all data points for that model. With these conditional posteriors in hand, we implemented collapsed Gibbs sampling by repeatedly cycling over each of the data points x_i, and sampling a new assignment z_i by computing each of the conditional posteriors $p(z_i = k)$ for each active component k.

Variational inference

Like MCMC, variational inference methods are tools for estimating parameters in posterior distributions over models for which exact learning is intractable. The following exposition of variational inference is drawn from Bishop 10.1 [15]. Consider a probabilistic model with observed variables $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and latent variables $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$. The model specifies the joint probability distribution $p(\mathbf{X}, \mathbf{Z})$, and our goal is to find an approximation for the posterior of the latent variables given the observed variables $p(\mathbf{Z}|\mathbf{X})$, and for the model evidence $p(\mathbf{X})$. For any distribution $q(Z)$ over the latent variables, we may decompose the log of the model evidence as follows:

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + KL(q\|p) \tag{2.50}$$

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln\left\{\frac{p(\mathbf{X},\mathbf{Z})}{q(\mathbf{Z})}\right\} d\mathbf{Z} \tag{2.51}$$

$$KL(q\|p) = -\int q(\mathbf{Z}) \ln\left\{\frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})}\right\} d\mathbf{Z} \tag{2.52}$$

where the summands of this decomposition define a lower bound on the log of the marginal likelihood, and the KL divergence of the distributions $p(\mathbf{Z}|\mathbf{X}), q(\mathbf{Z})$. Since the KL divergence is strictly non-negative, then if we drop the $KL(q\|p)$, we are left with an inequality which we may maximize with respect to the free distribution q. If we achieve equality, then we know that our estimate $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X})$. A further restriction imposed on the family of distributions q specifies how they decompose the full approximate posterior $q(\mathbf{Z})$ into factorized groups of latent variables $q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i)$. This is called *mean-field* approximation, again borrowed from physics. Mean-field variational approximation seeks to find the distribution which maximizes the variational lower bound $\mathcal{L}(q)$ as well as respecting a given factorization. Simplifying the notation by referring to $q_i(\mathbf{Z_i})$ as just q_i, any given factorization gives us:

$$\mathcal{L}(q) = \int \prod_i q_i \left\{\ln p(\mathbf{Z},\mathbf{X}) - \sum_i \ln q_i\right\} d\mathbf{Z} \tag{2.53}$$

$$= \int q_j \left\{\int \ln p(\mathbf{Z},\mathbf{X}) \prod_{i\neq j} q_i d\mathbf{Z}_i\right\} d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + c \tag{2.54}$$

$$= \int q_j \ln \tilde{p}(\mathbf{Z}_j,\mathbf{X}) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + c \tag{2.55}$$

so the distribution $\tilde{p}(\mathbf{Z}_j, \mathbf{X})$ is defined as the expectation $\mathrm{E}_{i\neq j}\left[\ln p(\mathbf{Z},\mathbf{X})\right] + c$ taken with respect to the q factor distributions over all variables $\mathbf{z}_i$ for which $i \neq j$. It is worth restating this expectation

$$\mathop{\mathrm{E}}_{i\neq j}\left[\ln p(\mathbf{Z},\mathbf{X})\right] = \int \ln p(\mathbf{Z},\mathbf{X}) \prod_{i\neq j} q_i d\mathbf{Z}_i \tag{2.56}$$

since the key insight into variational inference, is to consider *fixing* all factors $q_{i\neq j}$ via this expectation, and to then maximize $\mathcal{L}(q)$ with respect to all possible forms for the distribution q_j:

$$\mathcal{L}(q) = \int q_j \ln \tilde{p}(\mathbf{Z}_j,\mathbf{X}) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + c \tag{2.57}$$

$$= -KL(q_j \| \tilde{p}(\mathbf{Z}_j,\mathbf{X})) + c \tag{2.58}$$

which is minimized when they are the same distribution. The constant c is set by normalizing the optimal q_j^*. To restate, we achieve a general optimal expression for the variational approximation factors:

$$q_j^*(\mathbf{Z}_j) = \underset{i \neq j}{\mathrm{E}} [\ln p(\mathbf{Z}, \mathbf{X})] \tag{2.59}$$

This yields a recipe for finding an approximate solution to the posterior, depicted in algorithm 2.

Algorithm 2: Mean-field variational approximation example

```
input  : factorization of $q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i)$
output: lower bound $\mathcal{L}(q)$, variational distribution $q$
initialize values for $q_j(\mathbf{Z}_j)$  while not converged do
    for $j \in (1, \ldots, M)$ do
        compute $q_j^*(\mathbf{Z}_j) = \mathrm{E}_{i \neq j} [\ln p(\mathbf{Z}, \mathbf{X})]$
    end
end
```

Since it is likely that each of the q_j will depend on expectations computed with respect to the other factors, this algorithm is run iteratively, each time replacing the current estimate of q_j with new estimates according to equation 2.59. In the particular case of DPMMs, a variational approximation algorithm was developed using these principles by Jordan and Blei [16].

Memoized online variational inference One decided advantage of variational inference methods is that they optimize a lower bound on the model evidence, which is a readily interpretable measure of the goodness of fit. Model evidence may be estimated by MCMC, but this adds another layer of complexity to the problem. So to complement our own implementation of collapsed Gibbs sampling, we also employed a variational approximation algorithm for DPMMs by Hughes and Sudderth [53]. Their model constructs the DPMM by stick-breaking

$$p(\mathbf{x}, \mathbf{z}, \phi, v) = \prod_{n=1}^{N} F(x_n|\phi_{z_n}) Cat(z_n|w(v)) \prod_{k=1}^{\infty} Beta(v_k|1, \alpha_0) H(\phi_k|\lambda_0) \tag{2.60}$$

which is a process by which the mixing proportions v are explicitly generated. There is no advantage to collapsing any of the latent variables in variational inference, so the cluster indicators $\mathbf{z}$ and component parameters ϕ are all estimated. For our real valued data, we take F to be Gaussian, with prior base measure H as normal-Wishart. Their mean-field factorized variational distribution q takes the form

$$q(\mathbf{z}, v, \phi) = \prod_{n=1}^{N} q(z_n|\hat{r}_n) \prod_{k=1}^{K} q(v_k|\hat{\alpha}_1, \hat{\alpha}_0) q(\phi_k|\hat{\lambda}_k) \tag{2.61}$$

$$q(z_n) = Cat(z_n|\hat{r}_{n1}, \ldots, \hat{r}_{nK}), \quad q(v_k) = Beta(v_k|\hat{\alpha}_1, \hat{\alpha}_0), \quad q(\phi_k) = H(\phi_k|\hat{\lambda}_k) \tag{2.62}$$

Each of the hat-bearing hyper-parameters belong to the variational approximation q, rather than of the generative model p, though they have the same dimension. The evidence lower bound objective optimized is

$$\log p(\mathbf{x}|\alpha_0, \lambda_0) \geq \mathcal{L}(q) \triangleq \underset{q}{\mathrm{E}} [\log p(\mathbf{x}, v, \mathbf{z}, \phi|\alpha_0, \lambda_0 - \log q(v, \mathbf{z}, \phi)] \tag{2.63}$$

$\mathcal{L}(q)$ decomposes into a sum over several functions of the expected mass $\hat{N}_k$ and expected sufficient statistics $s_k(\mathbf{x})$ for each active component, as well as expectations of the parameters v, ϕ and estimates of the responsibilities $\hat{r}_{nk}$. See section 2 of [53] for more details.

2.6 Discussion

Analysis pipelines for image data from HCS are modular. Because they are so modular, there are many approaches to the different tasks that are not covered here. For other reviews of the field of high throughput imaging that focus on sample preparation and microscopy, see [72, 65, 78, 113]. For reviews that focus on image analysis, see [89, 34, 110], and for those focusing on data analysis see [109, 81].

Chapter 3

Detecting genetic involvement in DNA damage by supervised learning

This chapter presents our method to identify genes which have a functional connection to the repair of DNA damage. We begin by framing the problem of how to use high throughput screening to probe gene function, and reviewing the motivation and design for the experiments which produced the data analysed here, and in subsequent chapters. To help motivate the study, we present an overview of the two canonical pathways by which yeast recognizes and repairs DNA damage. We move on to describe the process of transforming cell images into feature data, follow that with our supervised learning method for cell classification, and our methods to score each sub-population. We then describe how to combine the results of different replicates, how to ensure sampling consistency, and how to guard against model over-fitting. We conclude with an analysis of high scoring genes in each condition, and contrast these with results from previous studies. At present, many HCS studies still score or evaluate populations by eye, which makes analysis difficult to reproduce, and expensive. Our method represents a way to reduce the amount of domain expert time, and enables screening on a much larger scale.

3.1 Inferring gene function from morphology

There is a long history in yeast genetics of determining gene function by looking at mutant phenotypes. The experimental design can be summarized as taking a population of yeast cells, dividing them up into genetically identical colonies, applying one unique deletion of a non-essential gene to each colony, and then observe the morphologies that arise in each. For example, one previous study of genetic interactions in yeast [28] used SGA to generate a wide array of gene-knockout mutant yeast[1] strains, with the aim of using the growth phenotype to characterize genetic interactions. As a proxy for fitness, the size and rate of colony growth were used to determine whether the deleted genes shared any role in different cellular processes. The coarse-granularity of colony size measurements overlooks many defined instances in budding yeast where colony growth is unremarkable in some single (or even double) mutant populations, even in the presence of significant morphological defects present within sub-cellular compartments. For example, nearly 50 percent of non-essential yest deletion mutants, while displaying no fitness defect, exhibit a number of morphological defects [88]. This substantial gap between genotype and phenotype

[1]in this thesis, yeast will be used as short-hand to denote the budding yeast Saccharomyces Cerevisiae

must be closed to further our understanding of how subtle events in cell biology impact complicated biological processes. Taken together, this presents a need for experiments designed to measure more complex sub-cellular phenotypes. One tool with the potential to recognize and understand sub-cellular phenotypes is high-throughput (HTP) multi-channel fluorescence microscopy. Genetic and drug perturbations can be applied to a large population of cells, and each cell can be captured in an image by high-throughput microscopy. These images are then processed to yield multidimensional datasets, which can be analyzed to associate specific genetic variations with observed phenotypes.

This study focused on using high content screening to find genes involved in the process of double-stranded DNA repair. In the budding yeast Saccharomyces cerevisiae, arrayed collections of haploid viable deletion mutants, as well as strains carrying conditional alleles of essential genes have allowed for extensive systematic characterization of the mechanisms that direct cellular processes.

3.2 Repairing double stranded DNA breaks

The response to DNA double-strand breaks (abbrv. DSBs) is a cellular process strongly conserved among eukaryotes. It involves a large number of checkpoint and repair proteins, usually working as part of larger complexes, which are responsible for damage detection, checkpoint activation, damage repair, and for resuming the cell cycle [69, 71, 8]. There are two pathways by which DSBs are repaired. The first is homologous recombination (abbrv. HR). The second is non-homologous end joining (abbrv. NHEJ). The choice of pathway is coordinated with the cell cycle: NHEJ is employed primarily in G1, while HR is used predominantly during both S-phase and G2/M [8]. Both NHEJ and HR follow stepwise pathways leading to repair, each of which uses a different set of proteins. Both pathways are rooted in DSB recognition by the MRX complex[2], which first recognizes and binds the exposed ends of the DSB.

DSB recognition In response to DSBs, a number of DNA checkpoint and repair proteins in S. cerevisiae including Rad51, Rad52, Rad53, Mre11, RP-A, Ddc1, Ddc2, Rad9, and Rad24 relocalize from a quasi-uniform nuclear distribution to distinct subnuclear chromosomal loci. DNA damage-induced foci co-localize with DSBs, and with regions of single-stranded DNA in vivo [69]. Foci are also observed at a low frequency in undamaged cells, which reflects the recruitment of repair proteins to spontaneous DNA damage [49]. Interestingly, these foci form preferentially in S/G2 phase suggesting that recombinational DNA repair is coupled to DNA replication.

Homologous recombination During HR, sequence information from a homologous DNA molecule is used as a template for restoring genetic information lost at the DSB. Recombinational repair of DSBs proceeds via a number of steps and involves proteins. First, the DSB is processed to yield 3'-single-stranded ends. These ends are bound by RP-A, which is presumed to protect the DNA against degradation and inhibit the formation of secondary structures. The subsequent invasion of the single-stranded DNA into homologous duplex DNA is catalyzed by the RecA homolog, Rad51, and stimulated by Rad52. The invading strand primes DNA synthesis of the homologous template, ultimately restoring genetic information disrupted at the DSB. Concurrent with recombinational repair, the DNA damage checkpoint is activated to slow DNA replication and arrest cells before cell division (G2 phase) until the DNA lesion has been repaired.

[2]Mre11, Rad50 and Xrs2

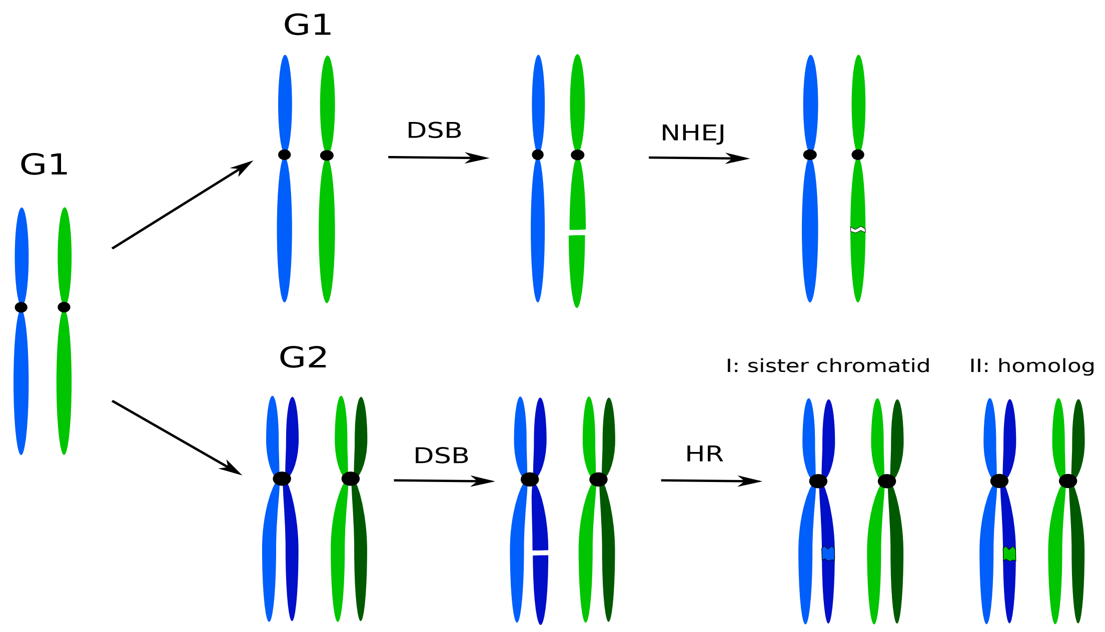


Figure 3.1: Re-drawn from [8], repair of a DSB by one of two pathways, according to cell cycle. The pathway chosen to repair a DSB is determined largely by cell cycle stage. In G1, cells have a single copy of each chromosome (light blue and light green). If a break occurs in G1, the cell repairs the DSB by NHEJ, directly re-joining the break in the chromosome (top). In G2, the chromosomes have been replicated, and so each has an identical copy of each original chromosome. DSBs in G2 are normally repaired by HR using either the sister chromatid or a homologous chromosome as a template (dark blue and dark green, bottom). Repair from the sister (I) results in restoration of the allelic information lost at the break site. Repair from the homologous chromosome (II) may lead to loss of heterozygosity, if accompanied by a crossover.

Non-homologous End Joining In non-homologous end joining, DSBs are repaired by directly ligating the double-stranded DNA at the locus of the break. The yeast Ku70/Ku80 heterodimer recognizes and binds the DNA ends. Ku70/Ku80 then recruits the Lif1/Nej1 heterodimer, and Lif1 both recruits and stimulates Dnl4 ligase activity to close the DSB.

3.2.1 RAD52 foci

The focus of the high throughput screen undertaken for this project was to discover genes that play a role in the HR pathway. In *Saccharomyces cerevisiae*, Rad52 is the defining member of the epistasis group[3] that includes several previously mentioned genes that have a role in HR such as Rad51, Rad55, Rad57, Dmc1, Rad59 and Mre11, Xrs2, Rad50. The Rad52 epistasis group is essential for HR [3]. Rad52's role is to bind single-stranded DNA and to motivate Rad51-catalyzed strand invasion. When DNA damage

[3]An epistasis group is a collection of genes that all operate within the same pathway.

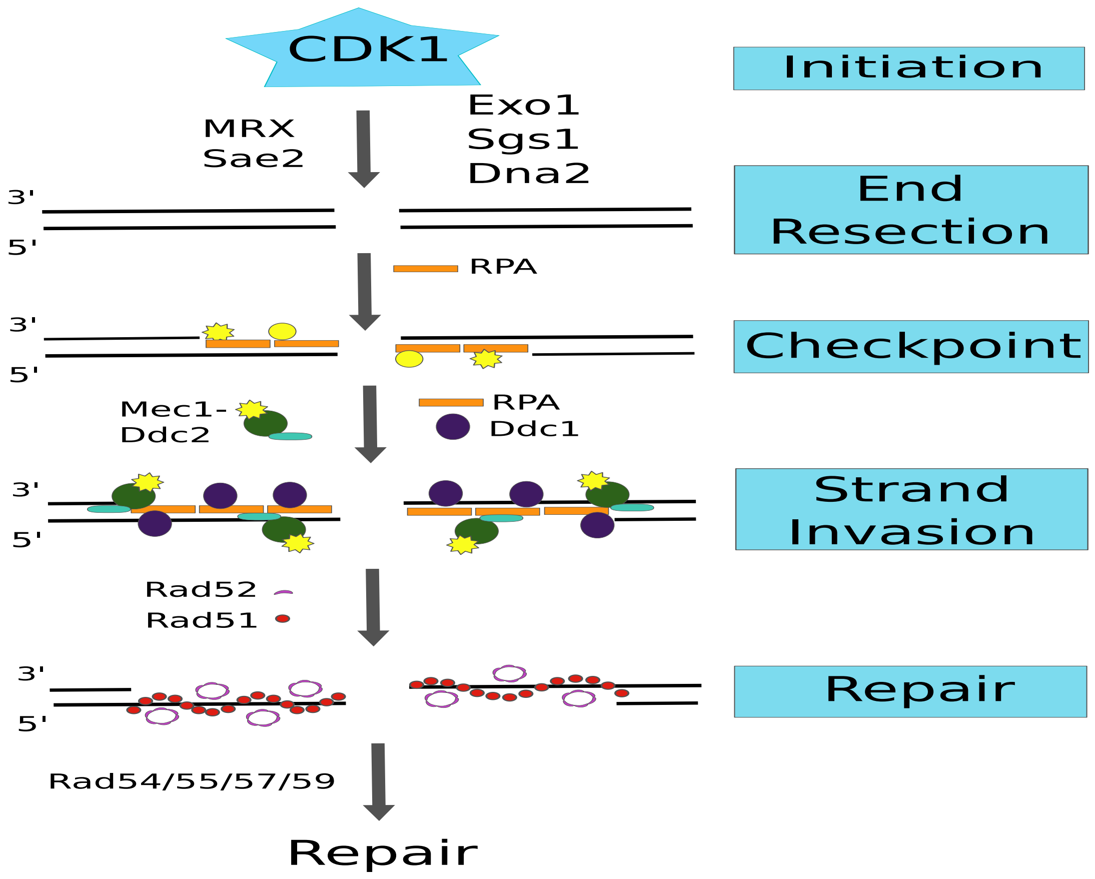


Figure 3.2: HR pathway diagram: recruitment of the HR machinery to a DSB is regulated (via phosphorylation) by both the major cell cycle kinase CDK1 and the checkpoint kinase Mec1. CDK1 phosphorylation of a protein is indicated with a yellow circle, Mec1 phosphorylation is indicated with a yellow star. When a DSB is detected, CDK1 recruits several nucleases to resect thhe 3' end of the DNA, forming single stranded DNA. RPA binds the single strands and recruits the Mec1/Ddc2 complex and the Ddc1/Mec3/Rad17 complex (just Ddc1 in the figure). Rad52 localizes to the resected ends, and catalyzes the formation of a Rad51 chain along the single stranded DNA. Finally, the HR process is engaged, closing the DSB.

is detected, proteins which play a role in HR relocalize to the break loci. Fluorescently tagged repair and checkpoint proteins have been used to explore the composition and interaction of different protein complexes that compose the foci. For the purpose of high content screening, fluorescently tagged Rad52 is a natural marker for studying the effects of gene knockouts on DSB repair, given the tight association between Rad52 foci and HR. Rad52 is required for the recruitment of all other HR proteins into repair foci [3]. It has been shown that mutants defective in various aspects of DNA metabolism, including damage checkpoints (Mec1 Sml1), HR (rad51D), and DNA replication (pol12100) exhibit elevated levels of spontaneous foci [69]. This elevation may be the consequence of an increased incidence of focus formation reflecting the generation of more DNA lesions, or the consequence of foci that persist. While the functional role of many proteins in DSB repair have been discovered using this marker [8], new roles for proteins continue to be uncovered, and more peripheral players remain uncharacterized. Taking into account the extent of genetic redundancy in *Saccharomyces cerevisiae*, this project aimed to discover these new players by performing whole genome screening in sensitized genetic backgrounds, where well-known proteins were specifically removed, or where the cells were placed under chemical stresses:

- Phleomycin is an an antibiotic compound produced by *Streptomyces verticillus.* Part of a family of antibiotics called bleomycin, it causes cell death in bacteria and fungi by inducing double-strand breaks of the DNA. It was used to create extra strain on the cells.

- The gene Sgs1 was deleted to perturb the HR pathway by removing a key member of the RecQ helicase-Topoisomerase III complex. It acts as a helicase, which along with the nucleases Exo1 and Dna2 processes the DNA at a DSB site post-detection.

- The gene Yku80 was deleted, to perturb the NHEJ pathway by removing one half of the Ku70/80 heterodimer complex.

3.3 Capturing the data

The experimental setup for this project combined two automated platforms to generate the image data. First, PCR mutagenesis was used to introduce fluorescent markers onto Rad52 (replaced with Rad52-GFP), HTA2 (replaced with HTA2-RFP), and RPL39 (replaced with RPL39pr-RFP). A query strain bearing these three mutations was crossed into the yeast deletion collection via synthetic genetic array (abbrv. SGA) [122], resulting in approximately 4450 different isogenic populations. Each population of cells was cultured to saturation in 96-well plates, and then sub-cultured to early log phase. Prior to imaging, cell culture was transferred to a well on 384 well glass slides. The Evotec Opera (PerkinElmer) automated spinning disk confocal microscopy system captured the images. Yeast cells were imaged live using a 60X objective, where 1 point on the Z-axis plane was automatically chosen, and an image was captured for 4-8 sites of each well. GFP and RFP signals were acquired simultaneously. The images were then saved to .tiff files, and processed into features vectors.

3.3.1 From images to features using CellProfiler

CellProfiler is a software package which provides a graphical user interface for performing analysis on biological images [20]. We devised and refined a pipeline to process each field of view image, to first isolate the cells and nuclei in each image, and then to transform each into a vector of features. Each screen consisted of approximately 23000 images, which were broken down into batches of 100 for batch processing in parallel to speed up computation.

Rescale intensities Both the red and green channel pixel intensities re-scaled to lie in $[0, 1]$.

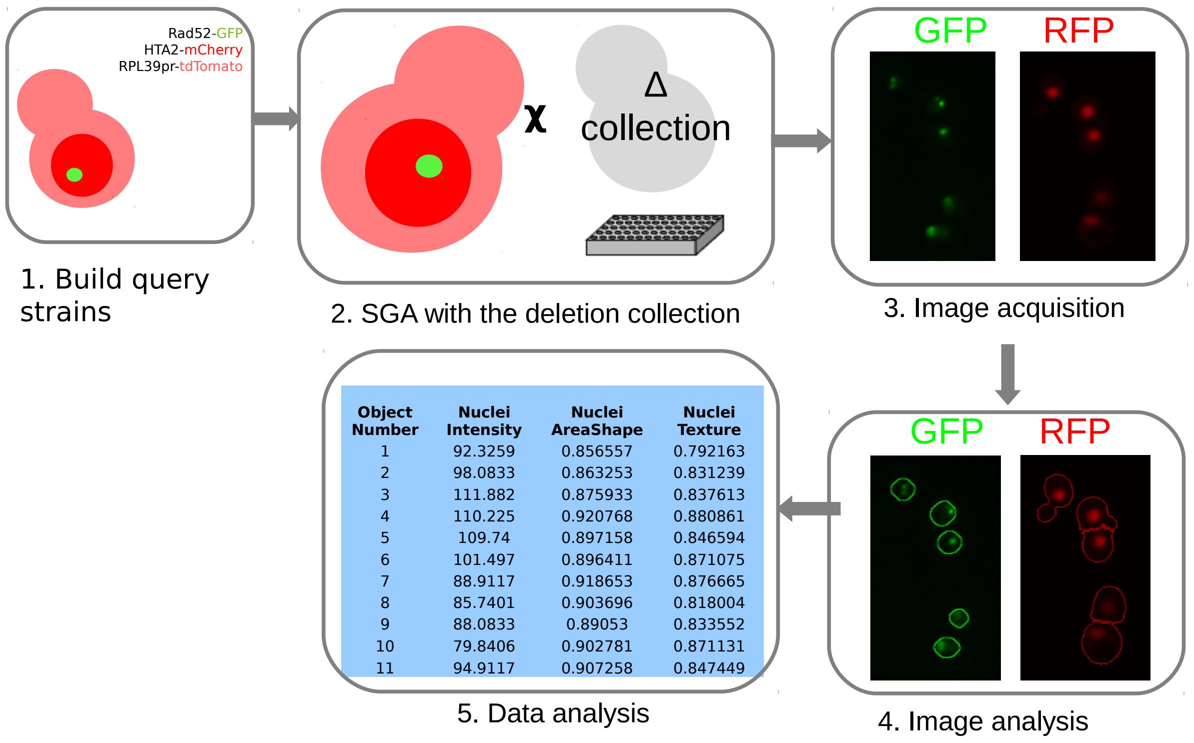
Segment nuclei Nuclei segmented by a fitting a mixture of two Gaussians model to the red channel intensities. This method assumes the pixel intensities are drawn from either a foreground or background distribution. Fitting the mixture recovers these Gaussians, which are used to calculate a threshold on the intensity values for labeling each pixel [104, 127]

Measure nuclei features Area, shape, intensity and textural features measured.

Segment cells Cells segmented using a seeded watershed algorithm variant, with nuclei as the seeds.

Measure cell features Area, shape, intensity and textural features measured.

After processing, each cell in each image was represented by a vector of 916 features.



Object Number	Nuclei Intensity	Nuclei AreaShape	Nuclei Texture
1	92.3259	0.856557	0.792163
2	98.0833	0.863253	0.831239
3	111.882	0.875933	0.837613
4	110.225	0.920768	0.880861
5	109.74	0.897158	0.846594
6	101.497	0.896411	0.871075
7	88.9117	0.918653	0.876665
8	85.7401	0.903696	0.818004
9	88.0833	0.89053	0.833552
10	79.8406	0.902781	0.871131
11	94.9117	0.907258	0.847449

Figure 3.3: A diagram of the experimental pipeline.

3.4 Classifying the cells

With feature vector representations for each cell in hand, we next address the problem of how to leverage a small amount of human labeled examples into a model that classifies the rest of the data of our screens, by labeling as yet unseen data. The nature of the experimental design and the complicated biology involved required us to make some simplifying assumptions.

- We chose to cast this problem as a binary classification. Though we observed some variation in the size and shape of foci, including some very rare instances of nuclei displaying multiple foci, we modelled cells using only two classes: cells either displayed a focus or they did not.

- As mentioned in (above), the current model of DSB repair via HR suggests that the presence of Rad52 foci is dependent on cell cycle stage. We chose not to specifically model cell cycle when classifying cells based on the presence of foci, due to the difficulty of accurately assigning cell cycle states to each cell.

- Foci are a transient phenotype; they arise due to breaks, but are repaired over the course of several minutes. We only had one exposure / image per population per condition, so we did not model the transient nature of the foci. This means we will be relying on having a large population of cells to offset any noise introduced by a random sampling time.

With these simplifying assumptions understood, we next cover the process of selecting informative features, of training the classifier, of interpreting the results, and finally of integrating results from different replicates into final lists.

3.4.1 Feature selection

We applied the Wilcoxon rank-sum (abbrv. WRS) test to determine the relevant features for our model. The features provided by CellProfiler are not drawn from a given probability distribution; some features were real-valued continuous variables (e.g cell or nuclear area), while others were restricted to specific intervals (e.g object eccentricity). A model-free test like the rank sum test could be applied to each feature without pre-processing or transformation.

A priori, we did not know which features made the most sense biologically, nor did we have the opportunity within CellProfiler to design any features. Thus a filtration style for feature selection which allowed us to fix the CP pipeline and then evaluate each feature afterwards seemed most appropriate. Features that showed no statistically observable difference between foci-bearing and non foci-bearing objects could be safely discarded. This had a useful benefit for the classifier trained on these features; support vector machines classify points based on similarity to the support vectors, so removing the nuisance features should lead to a better estimate of closeness to the relevant samples in the training set, and therefore to better SVM performance. Computing similarity over fewer features also provided a minor computational benefit.

One disadvantage of using the WRS is that it cannot take advantage of any correlations between features. This means features which might be highly correlated will be deemed significant, but which are so highly correlated that no new information is gained by adding them all; just picking a representative would suffice. In some extremely pathological cases circumstances, this may present a problem: say we wanted to select only the top X features as ranked by WRS p-values, and all the top X features happened to be highly correlated. We would be very vulnerable to any kinds of outliers in the test set, and our generalization performance would probably suffer.

3.4.2 Training the classifier

While there are many good models for binary classifiers to choose from, we used a Support Vector Machine (abbrv. SVM) in this instance. SVMs are powerful, flexible classification models that have been shown to excel on a wide variety of problems [37]. Like some more intricate models (e.g neural networks), they are able to learn nonlinear decision boundaries. Perhaps the best advantage of SVMs are that they employ a convex loss function, and so training is guaranteed to find the globally optimal parameter values.

Our training set was constructed by drawing cell image samples at random from a variety of mutant populations, and classifying them by visual inspection into positive (containing a focus) and negative categories. Each category consisted of approximately 1000 cells. We used CellProfiler Analyst, a tool for exploratory data analysis of high-throughput image-based experiments, to visualize and sort the cell thumbnail images [56].

We performed feature selection based on a Wilcoxon rank-sum test applied feature-wise (alpha $= 0.05$ with Bonferroni correction), which resulted in approximately 470 significant features. After discarding image and object number identification columns, the matrix containing the significant features of the training set were rescaled, so that each feature lay in $[0, 1]$. Each of the features we kept were positive real valued features, so this didn't drastically change their interpretation. After re-scaling, we used the libSVM library, with a MATLAB interface [21] to fit the model parameters.

We tested the performance of several different discriminative classification models, choosing eventually a simple linear kernel, whose hyper-parameters were fit using with 5-fold cross-validation and no additional regularization on the model parameters. Our results are depicted in Figure 3.4.

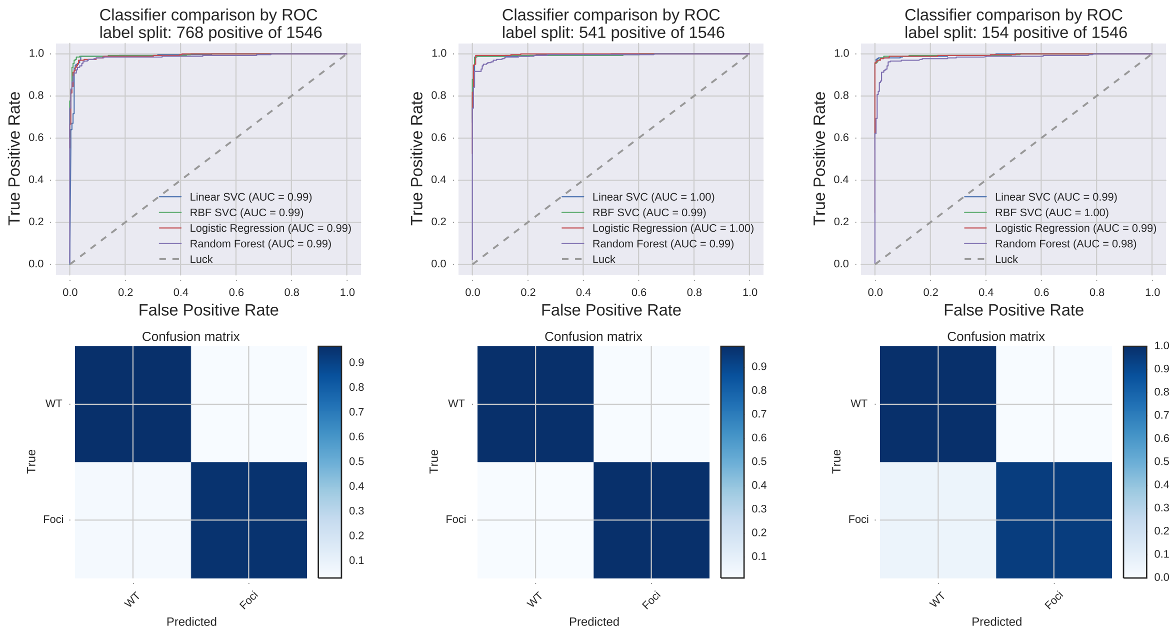


Figure 3.4: Foci classification using different models. Top panels are ROC curves comparing the discriminative classifiers labeled in the legends. Bottom panels are the confusion matrices for the linear kernel SVM on a test set. From left to right: the ratio of positive to negative examples used in the training (and test) sets is varied, and reported in the subtitle of the figures. The ROC of all four models is consistently high over all different training set splits, indicaitng that when predicting within a given replicate, the foci / no-foci ratio split does not significantly affect test performance.

As depicted in Figure 3.4, we experimented with different kernels, but found there was little room for improvement over a linear kernel SVM. As additional validation on the linear SVM classifier, we selected 50 images at random and examined the correlation between human assigned labels and computationally predicted labels. The strength of the observed correlation ($R^2 = 0.958$ see 3.5) was to convince us that our model was ready to be applied to the entirety of the unlabeled data.

3.4.3 Applying the classifier to a genome wide screen

Convinced that our model would allow us to accurately estimate the proportion of cells bearing foci in each population, we developed a pipeline to apply it to entire screens. The yeast deletion collection consists of 5376 genes, of which 4293 are compatible with the SGA process. While the number of cells observed in each isogenic mutant population varied widely, the sum total of mutant population sizes for each replicate was consistently on the order of 2 million cells. Each well on each plate was sampled at four different fields of view, resulting in over 21000 field of view images per replicate. Since each

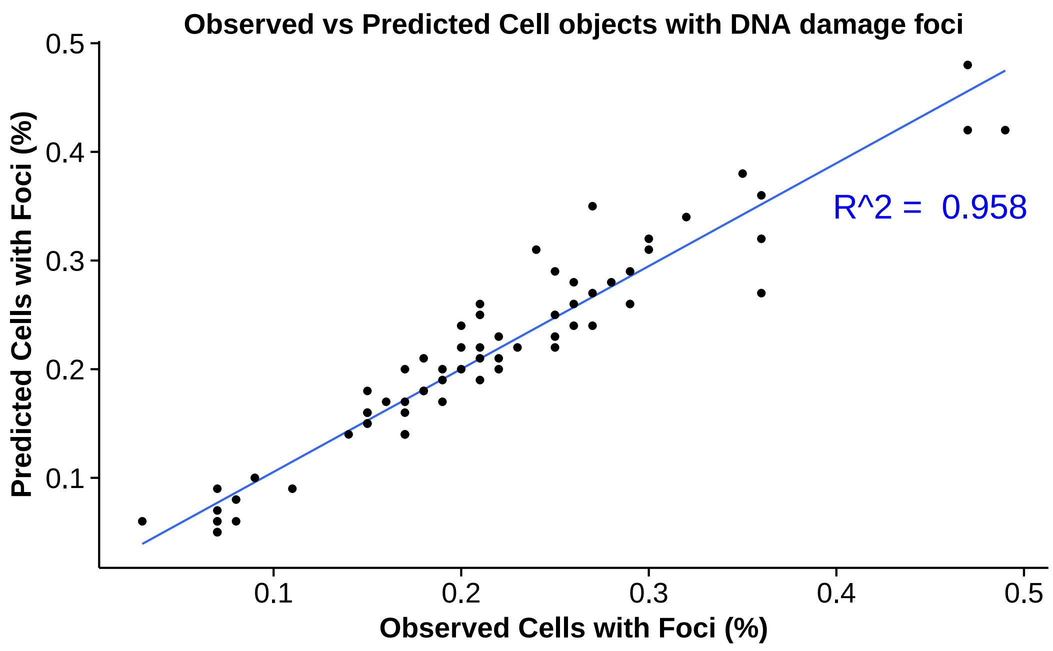


Figure 3.5: Scatter plot of observed percentage of cells with foci versus percentage of cells with foci predicted by linear SVC. For each of 68 sample images, an annotator counted the number of cells with foci appearing in the image. Divided by the total number of cells in the image produced a percentage of cells with foci. Similarly, our classifier predicted a label (foci / no foci) for each cell object in each sample image, which produces a percentage for each image. The blue line displays $y = x$, below which the Pearson correlation is displayed.

individual cell image can be evaluated without any dependencies on other cells, and so the problem of classifying batches of cells is trivially parallel. We distributed the evaluation across nodes in a shared filesystem cluster, partitioned the images into batches of 100. This allowed us to process one replicate of one screen in two to four hours, depending on the demand on cluster resources and Matlab licenses.

3.4.4 Scoring the populations

For each mutant population, we recorded the ratio of nuclei with foci to the total nuclei, defined as p_{obs}. Populations with fewer than 30 nuclei per replicate, or with fewer than 250 total nuclei from all combined experimental replicates were removed from further analysis. As we will discuss in section 3.4.4, any populations that do not exceed these thresholds lead to excessive uncertainty in our ability to estimate the percentage of the population displaying foci. All dubious ORFs in the collection were also removed. We examined the difference between each mutant population foci score and the score for a wild-type control population located on the same plate. For each mutant population, we calculated the binomial probability of observing m_{foci} from n_{nuclei} under the assumption that these observations were generated from the wild-type distribution parameter p_{wt}. Binomial test p-values were calculated for all mutants and all experimental replicates. The values for all replicates were combined using Fishers method to yield one final score. Let p_{ij} be the p-value calculated under the binomial for the j-th replicate of the i-th mutant population. Then the summed score over all replicates is calculated as

$$\chi^2_{ik} = -2\Sigma_{j=1}^{k} \ln(p_{ij}) \tag{3.1}$$

Below, we explore the different steps that have to be taken before transforming this raw significance score into information that we can reason about in the biological domain.

Examining the effect of sample size on population score

We observed some populations that showed a very small number of cells. How should we interpret the foci scores from these populations? Populations that had few were likely displaying synthetic sickness, but not necessarily due to dysfunction in DNA damage repair. As well, the number of observed foci can be affected by randomness introduced by the time of exposure, and the number of cells which happen to lie in the fields of view chosen.

To dig deeper into the relationship between the number of observed foci and the number of observed cells, we performed a bootstrap sampling experiment using three different populations. We sampled on two scales: first on a small scale ranging from 10 cells to 100 cells in increments of 10, and on a larger scale ranging from 150 cells to 2000 cells in increments of 50. We chose to sample from three populations (HIS3Δ , XRS2Δ and RAD51Δ), in three biological replicates. These populations were chosen to examine if the relationship is dependent upon the likelihood of DNA damage induced by genetic deletion. From inspection of Figure 3.2[4], populations bearing deletions of XRS2 or RAD51 are expected to show a higher proportion of cells exhibiting foci, while HIS3 deletions are expected to behave as wild-type. For each of the three gene pools, we sampled the designated number of cells randomly without replacement 100 times from populations of approximately 170,000 cells, and calculated the mean and standard deviation of the ratio of cells with foci to total cells sampled. We then took the average of both mean and standard deviation for foci ratio to estimate the mean and standard deviation for each sample size.

The results depicted in Figure 3.6 suggest that the uncertainty in the sample of cells exhibiting foci reaches a level around 1000 cells which remains constant up to 2000 cells, meaning we gain no further information when sampling over 1000 cells. We also see very noisy behaviour in the foci levels until we reach approximately 100 cells, meaning that we should be wary of populations with fewer than 100 cells sampled in total.

Correcting for positional and batch effects

In certain screens, we noticed that some high-scoring wells were proximal to other high-scoring wells on the plate. In some cases, plates had many more hits than expected under a random distribution of genes to wells (cf. Figure 3.7). The border wells of each plate in these experiments each contained HIS3Δ, which are expected to behave as wild-type yeast. Yet in plate five, depicted in Figure 3.7, the scores for these wells were elevated due to non-biological effects. In an experiment that is designed to detect genes of interest by finding outliers in the distribution over scores, it is important to discover and correct such errors.

There are several methods for correcting raw scores in high throughput screening assays. One method is to compare each the score x_i in each well relative to a set of positive and negative control wells C_h, C_l on the same plate [19]:

[4]and confirmed in previous studies [3, 49]

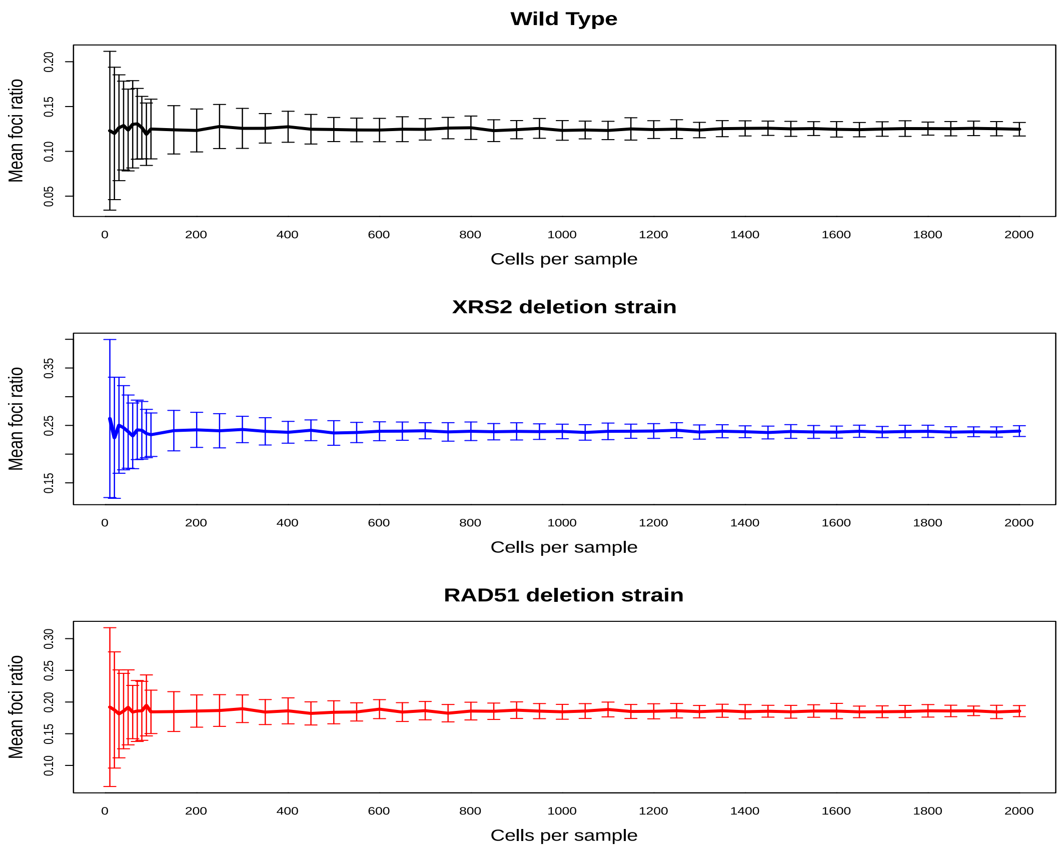


Figure 3.6: A figure displaying the bootstrap method used to determine effective population sample sizes for the foci phenotype. We sampled from three populations of yeast cells ($his3\Delta, xrs2\Delta$ and $rad51\Delta$), and in three biological replicates. We chose these three to examine if the relationship is dependent upon the likelihood of DNA damage induced by genetic deletion. Both XRS2, and RAD51 deletions are expected to show higher proportion of cells exhibiting foci as they are part of the canonical pathway for DNA damage repair by homologous recombination, while HIS3 deletions are expected to behave as wild type.

$$\frac{C_h - x_i}{C_h - C_l} \times 100 \tag{3.2}$$

While commonplace, this suffers from several deficiencies. It does not address systematic positional variability, such as edge effects or plate effects. Unwanted variability can be introduced depending on the size of each control sets, as well as by outliers within each set. A popular alternative from statistics is the Z-score transformation:

$$z_i = \frac{x_i - \bar{x}}{S_x} \tag{3.3}$$

where the score x_i for the population in well i is transformed into z_i by subtracting $\bar{x}$ (the empirical mean of the plate), and dividing by S_x, the standard deviation of the plate. This has several advantages over control-based normalization. It accounts for plate variation, and uses the distribution of scores over the entire plate as a baseline against which to measure the significance of the score in well i. However, it still fails to correct for any positional effects on the plate. As well, both the empirical mean and standard deviation can be skewed by the presence of outliers on the plate. For example, a few very high well

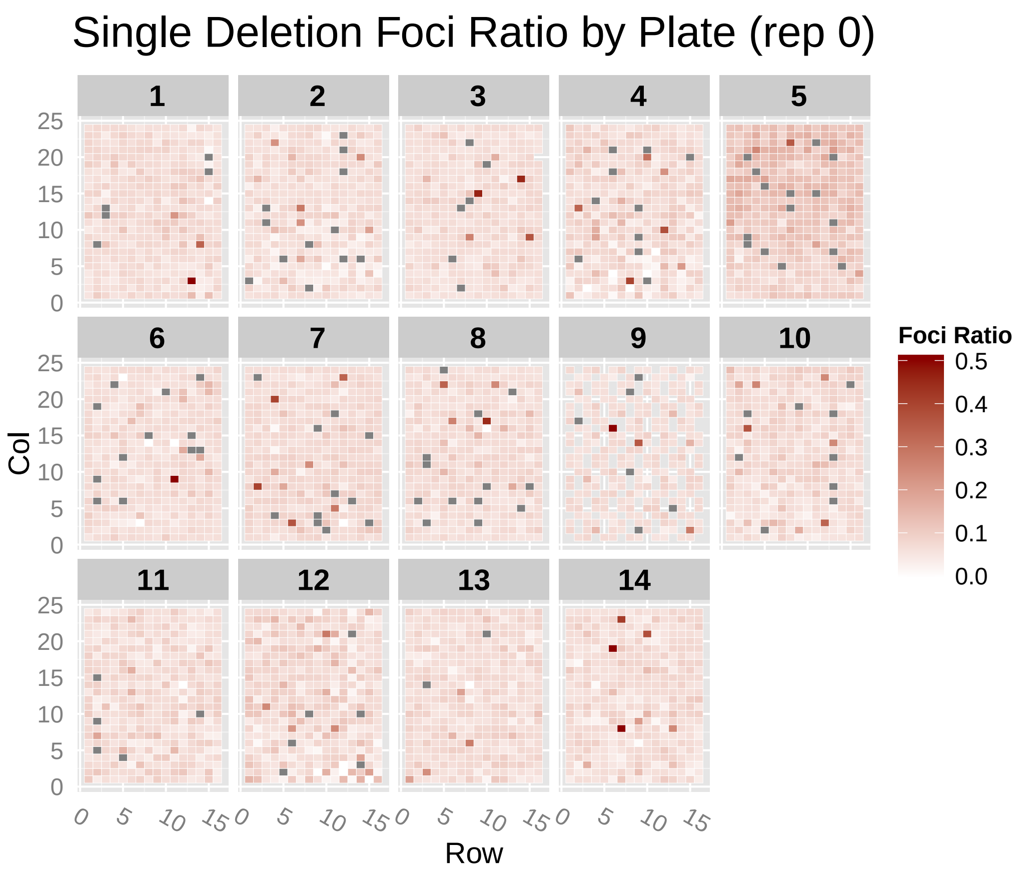


Figure 3.7: Heatmap of raw foci ratio values (number of cells with foci divided by total observed cells). The border of each plate in the screen contains HIS3 deletions, which are expected to behave as wild-type. The border wells of plate 5 show elevated levels of cells with foci, which is not biologically plausible.

scores will push moderately high well scores towards a Z-score of 0, while a few very low well scores will inflate otherwise unremarkable scores. The B-score [19] resolves both of these weaknesses. This model, specific to data that lies on a 2-d grid, fits an additive model to the raw score x_{ijp}

$$\begin{aligned} x_{ijp} &= \mu_p + \rho_{ip} + \gamma_{jp} + \varepsilon_{ijp} \\ r_{ijp} &= x_{ijp} - \hat{\mu_p} - \hat{\rho_{ip}} - \hat{\gamma_{jp}} \\ b_{ijp} &= \frac{r_{ijp}}{mad_p} \end{aligned}$$

Here, x_{ijp} is the raw value of the well in the ith row, jth column on the pth plate. μ_p models the contribution to x_{ijp} by the plate, ρ_{ip} and γ_{jp} model contributions from the row and column, while ε_{ijp} models the random effect of the score contributed by the population including measurement noise. The hatted versions of these represent the values that are inferred through Tukey's two-way median polish [19]. The numerator after adjustment, r_{ijp}, subtracts estimated contributions from the plate, row and column from the observed score. This allows it to account for positional effects. The denominator, mad_p, is the median of absolute deviations for the residual scores r_{ijp}. The idea here is that once the fixed effects from r scores should resemble random noise centred at 0, assuming there are no high scoring populations on this plate. The absolute values of the estimated ε_{ijp} values, called the absolute deviations, measure the spread of the distribution of r_{ijp} values. Dividing each estimated residual by the median of the absolute deviations will allow moderate hits to stand out as such, without being influenced by a handful of outliers.

Comparing scoring mechanisms by precision

We have described several methods for normalizing the raw scores, and also for combining the binomial test p-values into one final score using Fisher's method. Next, we will discuss how these two components were joined in a pipeline to produce a ranked list of genes. We chose not to perform tests for significance on the combined Fisher scores, the χ^2 values produced by Fisher's method, preferring instead to use precision as a measure of appropriate ranking. Choosing a method with highest precision seemed more attractive, since the results had a biological interpretation. We could compare how well different ordered subsets of our ranked list of genes reflected a list composed of DNA damage genes drawn from recent literature. We compared four different scoring methods, based on the precision measured against a gold standard list:

- Binomial distribution p-value score based on the wild-type populations on each plate, filtered by B-score
- Binomial distribution p-value score based on the wild-type populations on each plate
- Z-scores of binomial distribution p-value score based on the average percent foci of all wild-type populations in the screen
- Binomial distribution p-value score based on the average percent foci of all the wild-type populations in the screen

B-scores from different replicates were averaged together to yield one final B-score. Mutant populations with a B-score of less than 1.5 were filtered out from the final hit list. Combining the binomial distribution test normalized to plate-specific wildtype averages of percent foci with a B-score filter is the most adept at eliminating false positive hits. Based on these criteria, mutants with a final Fishers score greater than or equal to 40 (non-essential genes) and 400 (essential genes) were deemed hits.

3.5 Results

Applying the pipeline as described in sections 3.4,3.4.2 and 3.4.4 produced a list of genes for each condition ranked by the adjusted score, aggregated over replicates. From each non-essential experimental condition, we selected the top 50 hits and depicted them as a network using GeneMANIA [129], a web-tool for visualizing gene networks[5] They are each depicted in subsections below. Each node in each figure is coloured according to the GO biological process for which it is most enriched. The edges between the nodes indicate either physical (i.e protein-protein) interaction, or genetic interaction. Below, we discuss the results were largely consistent across conditions, and contrast our results with a previous study by Alvaro et al. As well, we present select surprising hits that are not readily explained by existing knowledge, and are candidates for more experiments.

[5]GeneMANIA is actually a network fusion method which, given a set of genes and desired interaction datasets, finds other genes that are related to a set of input genes. Though it provides a very handy visualization interface all the same.

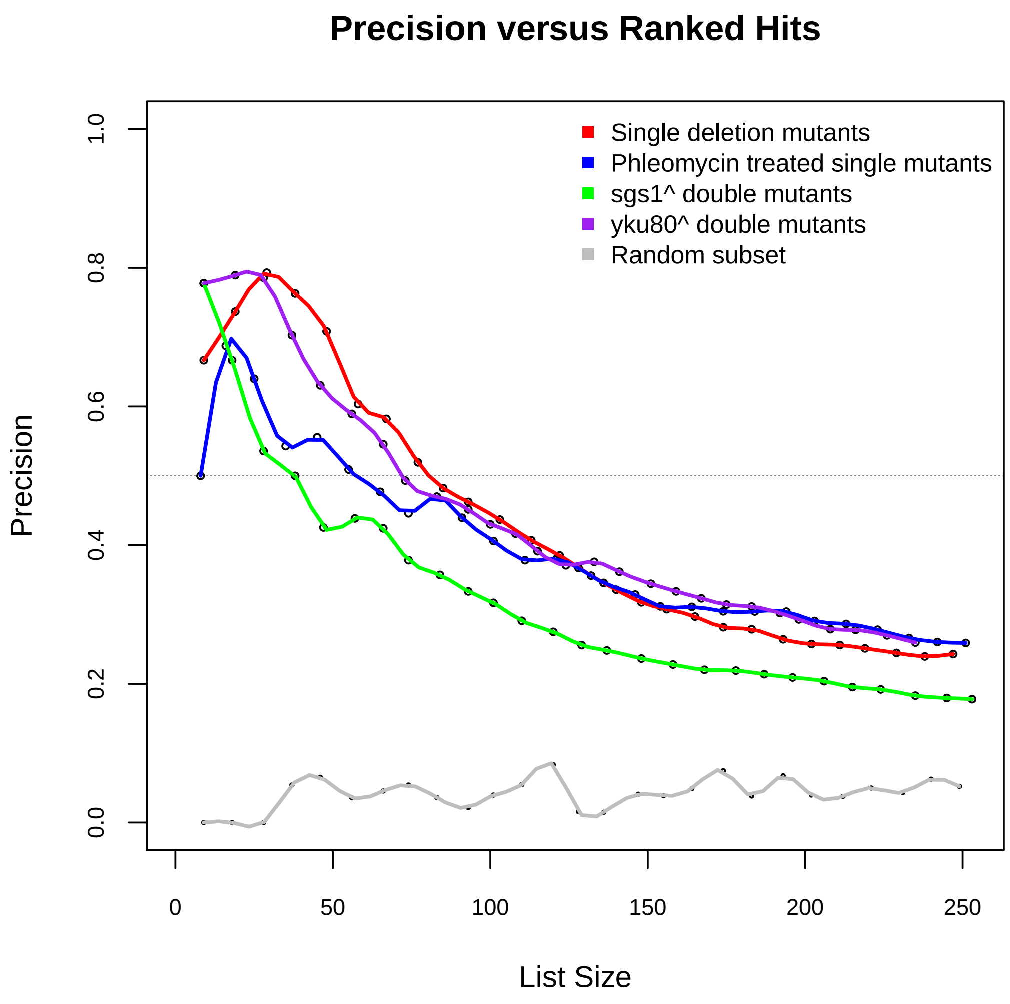


Figure 3.8: Precision curves for the aggregated results of different experimental conditions against annotated gold standard lists of genes implicated in DNA damage repair.

3.5.1 Comparison with Alvaro et al.

The study by Alvaro et al. produced a set of 116 genes which were shown to be involved in the repair of DNA damage, and here we overlay their results with our own in Figure 3.9. The Alvaro study used a different measure of signifiance than our study, which makes the two difficult to compare. A gene in their study had to exceed a threshold of a 4 fold change over wild-type to qualify as a hit. A gene in our study had to exceed a combined Fisher's method score that was set according to a level calibrated by acceptable loss of precision (c.f section 3.4.4). There are also significant differences in sample sizes: the median population size in the Alvaro study was 230 cells total, whereas our study had a median population size of 966[6]. Their scoring was performed by eye, where as our scoring was performed by an automated pipeline. These combined differences may account for the discrepancy in overlapping results. While our scoring is likely to be more consistent due to the lack of human error in applying

[6]combining biological replicates.

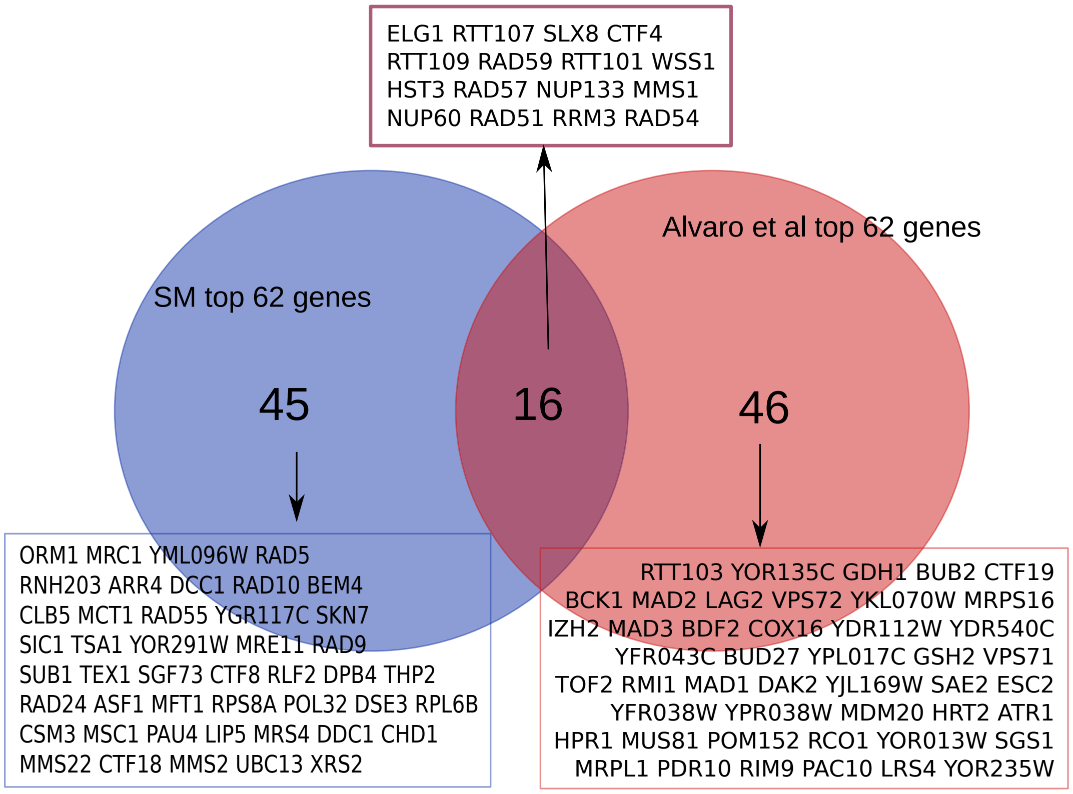

ELG1 RTT107 SLX8 CTF4
RTT109 RAD59 RTT101 WSS1
HST3 RAD57 NUP133 MMS1
NUP60 RAD51 RRM3 RAD54
Alvaro et al top 62 genes
SM top 62 genes
45
16
46
ORM1 MRC1 YML096W RAD5
RNH203 ARR4 DCC1 RAD10 BEM4
CLB5 MCT1 RAD55 YGR117C SKN7
SIC1 TSA1 YOR291W MRE11 RAD9
SUB1 TEX1 SGF73 CTF8 RLF2 DPB4 THP2
RAD24 ASF1 MFT1 RPS8A POL32 DSE3 RPL6B
CSM3 MSC1 PAU4 LIP5 MRS4 DDC1 CHD1
MMS22 CTF18 MMS2 UBC13 XRS2
RTT103 YOR135C GDH1 BUB2 CTF19
BCK1 MAD2 LAG2 VPS72 YKL070W MRPS16
IZH2 MAD3 BDF2 COX16 YDR112W YDR540C
YFR043C BUD27 YPL017C GSH2 VPS71
TOF2 RMI1 MAD1 DAK2 YJL169W SAE2 ESC2
YFR038W YPR038W MDM20 HRT2 ATR1
HPR1 MUS81 POM152 RCO1 YOR013W SGS1
MRPL1 PDR10 RIM9 PAC10 LRS4 YOR235W

Figure 3.9: Venn diagram of the 62 confirmed hits of the Alvaro et al. study versus our own. We took the 116 genes specified as hits for the Alvaro et al. study, which were genes which displayed more than 20% of cells with a focus. We then filtered these by admitting only those that repeated this in one of two replicate experiments, which resulted in 62 genes (c.f Supplementary Data [3]). For our own data, we chose the top scoring 62 genes in the SM study.

phenotype labels to cells, the accuracy may be lower than a human expert (Founk & Styles, personal communication). Another key difference, is that genes in the Alvaro study had to score highly in the initial screen to be considered a hit; our screen was agnostic to the ordering of the biological replicates, allowing for more potential hits to be discovered after aggregating all results.

3.5.2 Recovery of genes to be hits

In each of the different conditions, there are groups of genes deemed hits by our study which are readily explained based on the existing literature of genes exhibiting DNA damage foci. All conditions displayed significant enrichment for genes that are annotated as participating in germane GO biological processes of DNA repair, response to DNA damage stimulus, double-strand break repair, and recombinational repair. Deleting genes directly involved in the prevention of DNA damage would be expected to display increased levels of foci. What this study and others show is that that Rad52 foci can result from many different sources of cell damage.

Single deletion mutants

Beyond the GO categories listed in Figure 3.10, the top categories for GO biological process that showed enrichment in the top 100 hits for the single mutants were: nuclear division (4.31E-10), organelle fission (6.45E-10), DNA replication (2.07E-8), meiotic cell cycle (1.82E-6), mitotic sister chromatid cohesion (3.98E-6), and telomere maintenance (1.00E-5). Particularly interesting that the gene TEX1 appears in the top 50 hits. TEX1 (YNL253W) is involved in mRNA export as a component of the transcription export (TREX) complex. While its annotated function is not directly related to DNA damage repair, our results suggest it may have a peripheral or complimentary role to play in DNA damage repair yet to be fully characterized.

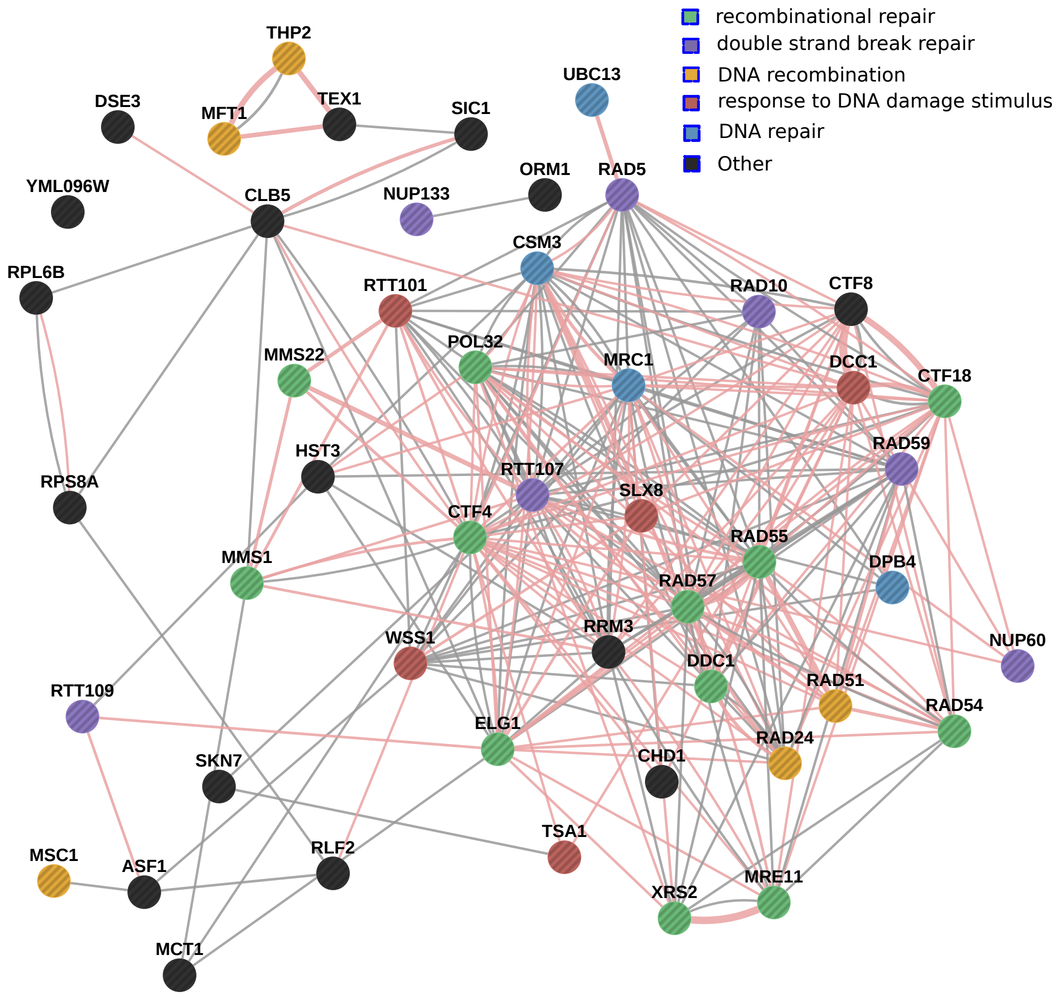


Figure 3.10: Single deletion mutants, top 50 hits. Nodes are coloured by GO biological process for which they are most enriched (depicted in figure legend). Links between nodes indicate evidence in the literature for physical (protein-protein) interaction, or for genetic interaction.

Single deletion mutants with phleomycin treatment

Single mutants with elevated foci in the presence of phleomycin, an exogenous source of damage, included genes that likely play a role in protecting the cell against outside sources of damage caused by environmental contaminants such as drugs. The top categories for GO biological process that showed enrichment in the top 100 hits for phleomycin were: organelle fission (1.12E-6), nuclear division (3.42E-6), retrograde transport, endosome to Golgi (1.08E-5), maintenance of DNA repeat elements (1.11E-4), DNA integrity checkpoint (1.11E-4), DNA damage checkpoint (1.11E-4) and meiotic cell cycle (1.50E-4).

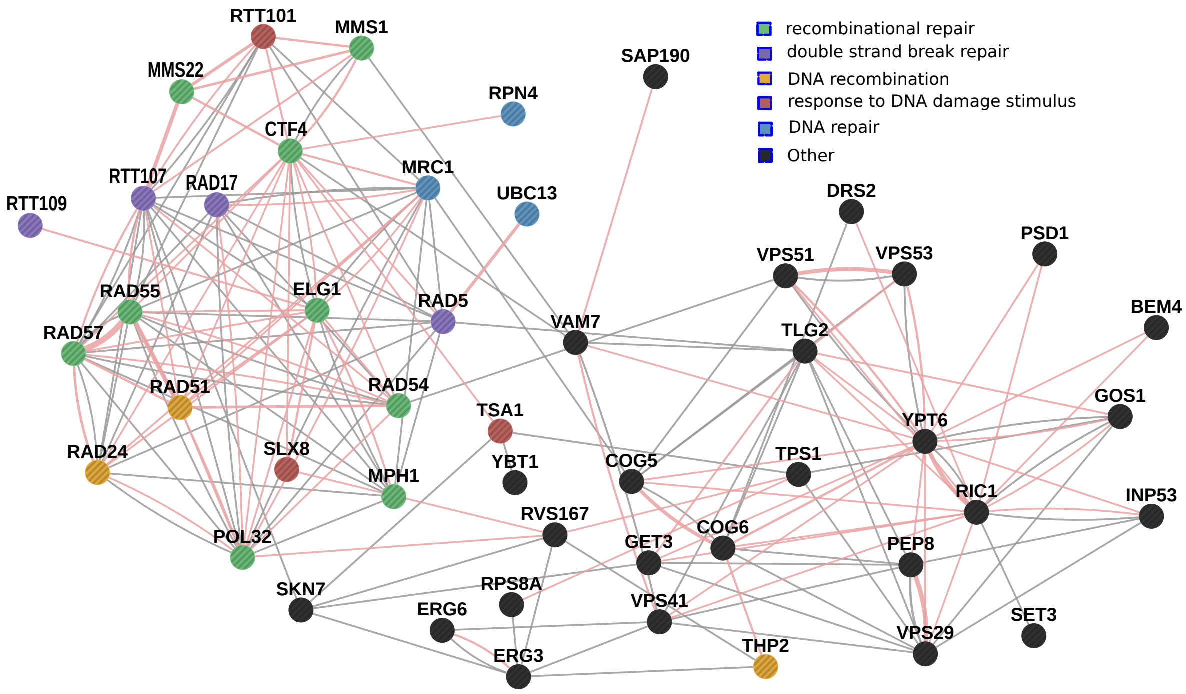


Figure 3.11: Phleomycin + deletion mutants, top 50 hits. Nodes are coloured by GO biological process for which they are most enriched (depicted in figure legend). Links between nodes indicate evidence in the literature for physical (protein-protein) interaction, or for genetic interaction.

Unsurprisingly, many of the same GO categories seen in the single mutant condition alone are represented again here. Of special note are the appearance of transport based genes VPS29,VPS53,VPS51,VPS41. A chemical analog of phleomycin (bleomycin) is known to travel via endosomal transport for storage in vacuoles, whether for eventual degradation or secretion via the Golgi [5]. If this transport mechanism is inhibited, the drug remains in the cytoplasm, where it may diffuse across the nuclear membrane and degrade DNA.

Double mutants with SGS1 deletion

For the sensitized background with single mutants coupled with a deletion of SGS1, the top categories for GO biological process that showed enrichment in the top 100 hits were: mitotic recombination (2.97E-4), telomere maintenance (5.10E-4), telomere organization (5.17E-4), anatomical structure homeostasis (5.10E-4), DNA replication (1.19E-3), DNA-dependent DNA replication (7.05E-3), chromatin organization (7.05E-3) and DNA strand elongation (7.29E-3).

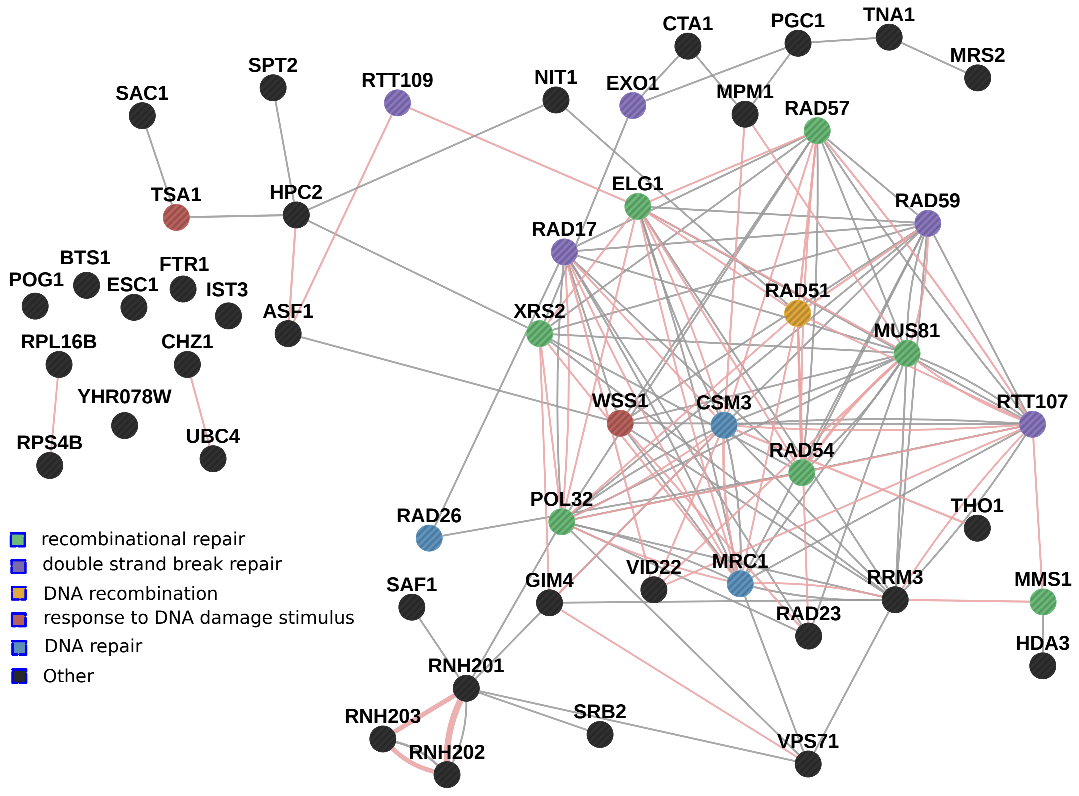


Figure 3.12: Double mutants with SGS1 deletion, top 50 hits. Nodes are coloured by GO biological process for which they are most enriched (depicted in figure legend). Links between nodes indicate evidence in the literature for physical (protein-protein) interaction, or for genetic interaction.

The deletion of SGS1 results in a serious challenge for the cell to recover from DNA damage and errors in replication. Because of low population sizes due to synthetic sickness by SGS1 deletion, many strains required extra replicates to ensure an aggregate sample size in line with the other conditions. We see several genes that ranked in our top 50 that appear only in this context, which can be interpreted as a benefit of screening in many different genetic backgrounds. Of the disconnected singleton genes in Figure 3.12, interesting standouts include uncharacterized ORF YHR078W, as well as POG1 and FTR1. YHR078W has no known function, though the presence of predicted transmembrane domains suggest it may be a membrane protein. According to the Saccharomyces Genome Database, POG1 acts is a nuclear chromatin-associated protein of unknown function that may have a role in cell cycle regulation. FTR1 is an iron permease, involved in the transport of iron across the plasma membrane whose expression has been noted to increase in the presence of DNA replicative stress [23].

Double mutants with YKU80 deletion

For the sensitized background with single mutants coupled with a deletion of SGS1, the top categories for GO biological process that showed enrichment in the top 100 hits were: regulation of DNA metabolic process (4.41E-11), nuclear chromosome (1.43E-10), telomere maintenance (4.18E-10), anatomical structure homeostasis (4.18E-10), telomere organization (4.78E-10), maintenance of DNA repeat elements (3.85E-8), double-strand break repair via homologous recombination (4.75E-8), chromatin modification

(9.88E-7), histone modification (1.61E-6) and non-recombinational repair (2.28E-6).

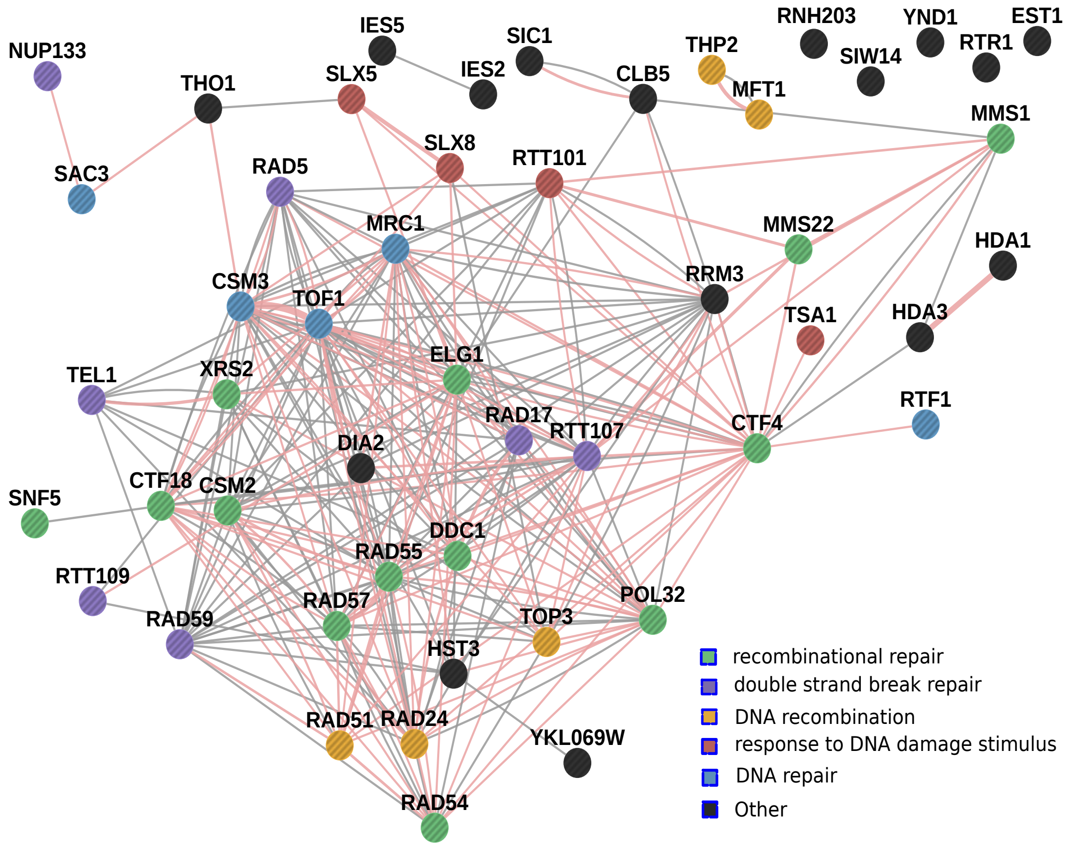


Figure 3.13: Double mutants with YKU80 deletion, top 50 hits. Nodes are coloured by GO biological process for which they are most enriched (depicted in figure legend). Links between nodes indicate evidence in the literature for physical (protein-protein) interaction, or for genetic interaction.

This is only a handful of the myriad functions for which the top 100 genes of the YKU80 hit list is enriched. Interestingly, it suggests not only is the cell struggling to repair DSBs by NHEJ (viz. the enrichment for non-recombinational repair), but also by homologous recombination. Several noteworthy singletons in the top 50 hit graph 3.13 are RNH203, which is required for effective RNase H2 activity, and thus for the RNase H2 complex's role in lagging strand DNA synthesis which occurs as part of NHEJ. Another is YKL069W, a methionine-R-sulfoxide reductase that has been shown to localize to the nucleus in conditions of DNA replicative stress.

3.6 Discussion

In this chapter, we have described a series of methods working in a pipeline to process high-content screening images. Together, these methods take cell objects represented as a bag of features with a small subset of labels, and produce a ranked list of genes whose order reflects their importance in the process of repairing DNA damage. We described a genome wide study of genes that play a role in repairing double-stranded DNA damage foci in yeast, using high content screening and image analysis. We classified all cells in the study into those which did (or did not) display a specific DNA damage

phenotype using a support vector machine, and how we used those predicted labels to derive a score assigned to each gene. Finally, we combined the results from replicate screens to produce a ranked list of genes, allowing us to identify which genes merited further study.
The major contribution of this work was determining the proportion of cells with observed DNA damage foci in cell images by casting it into a binary classification model. Also important were the application of statistical methods for estimating the reliability of results with respect to sample sizes, for normalizing the raw scores, and for combining different replicates into one final ranking of genes.

3.6.1 A word about performance on different replicates

We have shown that when aggregated over several biological replicates, our method produces the desired results, in the form of a ranked list of genes primarily associated with biological processes relevant to DNA damage. However, this does not tell us about the performance on each individual replicate. We investigated how the performance of our classifiers varied across biological replicates by measuring the ROC (cf. Figure 3.4), but contrast the results obtained on the first replicate of the single deletion condition to that of a different replicate:

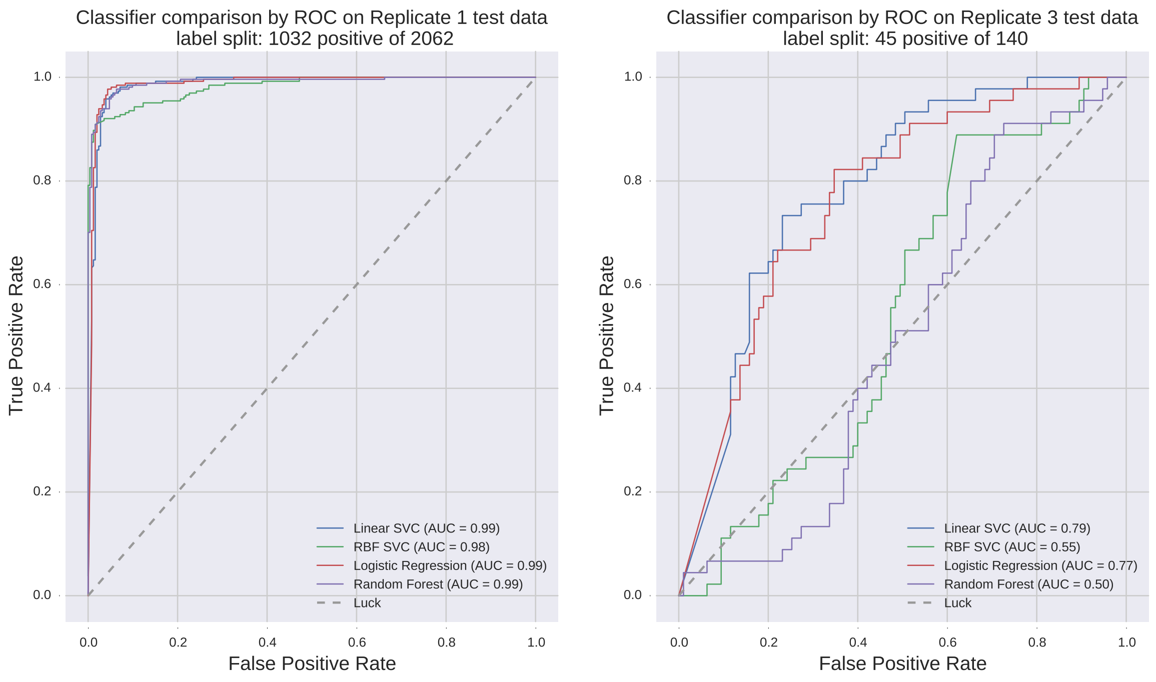


Figure 3.14: Foci classification using different models, as measured by ROC. Left is the test set from replicate one of the single deletion condition. Right is a test set drawn from replicate three of the single deletion condition. The ratio of positive to negative examples used in the test sets appears in the subtitle of each panel.

Inspection of Figure 3.14 reveals that each classifier which was trained on data from replicate one has diminished performance when predicting labels on data from replicate three. The consistent behaviour of each of these different classifiers suggests some fundamental difference between the two datasets.

If we consider data from replicate one as training data and data from replicate three as test data, the observed differences in ROC performance of the classifiers might also be explained by a phenomenon known as covariate shift. In statistics *covariate shift* in a classification (or regression) problem is defined as a discrepancy between the marginal densities over covariates of both sets, yet no difference in the class-conditional densities:

Covariate shift occurs when $P_{tr}(y|x) = P_{tst}(y|x)$ but $P_{tr}(x) \neq P_{tst}(x)$.

Covariate shift may also be interpreted as a violation of the assumption that data from training and test sets follow the same distributions [111, 80]. We will investigate this possibility further in chapter 6.

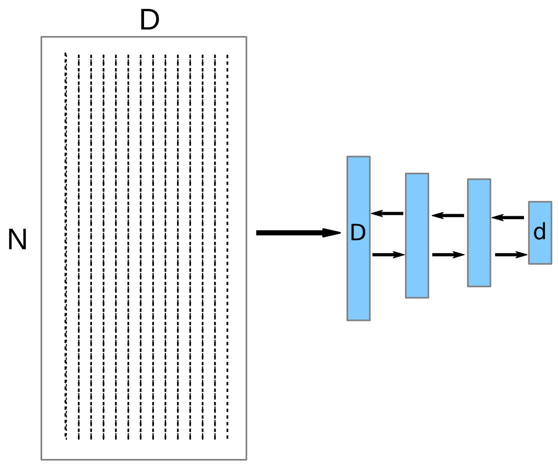
Chapter 4

Dimensionality reduction using deep autoencoders

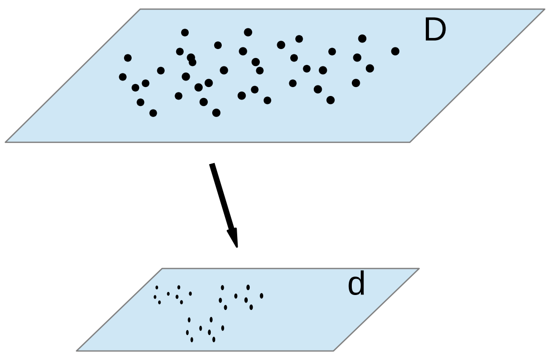
In the previous chapter, we described our method to classify cells in different populations, and to use the predicted label proportions of each population as evidence that the gene deleted from its cells played a role in double stranded DNA damage repair. That project built on the considerable body of prior biological knowledge about DNA damage foci. Leveraging what we knew about the formation of RAD52 at sites of double stranded break repair, we could measure the proportion of cells that displayed a well-defined phenotype, and convert this into precise indicator of functional involvement in DNA damage. We could also guess that certain mutations would make it difficult for cells to repair DNA damage. Those populations would be enriched for cells displaying the focus phenotype, and were good candidates to mine for labeled examples. This leads us to two assumptions upon which studies that employ fluorescence based reporter tagging for organelle morphology [126, 38, 99] and gene function rest. First, there must be a sufficient understanding of the pathways involved to identify and tag a candidate gene to reveal the variation in morphology. Second, there should be sufficient knowledge of positive controls to facilitate the collection of phenotype exemplars, so that models can be trained to identify and label the remaining cells in the experiment. However, it will not always be the case that positive controls are available to be mined. Even when positive controls are identified, there are some phenotypes which have very low penetrance even among affected populations. Others phenotypes are transient, and may be difficult to identify reliably. In the absence of positive control populations to mine for examples, we must resort to random sampling of images to identify phenotypes by eye, which is extremely time inefficient.

Though we may no longer be able to rely upon labeled data, our goal remains the same: to discover genetic deletions that lead to a loss of function that is detectable by distribution over phenotypes. One alternate way to do this is to characterize populations as either being alike or unlike a wild-type population. If we assume that most genetically altered populations will not display vast differences in the distribution over phenotypes as compared to wild-type populations, then we may reduce the search for different proportions of phenotypes into unsupervised outlier detection. This may be an advance, but it is complicated by the high-dimensional representation chosen for our data. In order to facilitate outlier detection, we need a more compact representation to our data.

In this chapter, we present a method for dimensionality reduction of a very large, high dimensional dataset, where the quality of the output is measured by how well cells with similar phenotypes are



(a) Schematic diagram of how the model, a deep-autoencoder, reduces the data from dimension D to d, where $D >> d$.



(b) Schematic diagram of embedding the data into a lower dimensional space.

Figure 4.1: A diagram of our method for dimensionality reduction for high content screening data. In the first stage (a), the data is used to construct and train a deep-autoencoder that learns to reproduce the data after compressing it. We then apply the model to the entirety of the dataset (b), resulting in a lower dimensional embedding of the data.

clustered together in the lower dimensional space. We begin by describing the problem, and show how conventional methods for dimensionality reduction are ill-suited to the attributes of our dataset. We then describe how to overcome these deficiencies by constructing a deep autoencoder model, and evaluate our model against comparable methods on a smaller validation dataset.

4.1 Dimensionality reduction for high dimensional data

Before we describe our method, we will take a step back and consider why we want to do dimensionality reduction in the first place. Our end goal is to fit a particular statistical model to our dataset, one that will allow us to detect outliers. Previous attempts in the context of high-content screening have used mixture of Gaussian distributions to model the joint density over features [126, 112]. In both cases, the researchers chose to transform the data into low-dimensional features prior to fitting the mixture model, a choice motivated by a phenomenon dubbed the curse of dimensionality [48]. The curse of dimensionality is an observation that as the dimensionality D of a space increases, the amount of data needed to fill all the subspaces increases at a rate exponential to that of $|D|$. This problem manifests itself in a slightly different way when clustering high dimensional points in space based on distance. As dimensionality increases, the difference between the supremum over pairwise distances and the infemum over pairwise distances shrinks. Mixture model distributions assume that proximal points in a given space were generated by the same component in the mixture. Therefore, to effectively estimate the distribution that generated the observed data, distances between points generated by the same component should be proximal, while those generated by different components should be distal. The curse of dimensionality means the difference between these distances becomes small in high dimensions. For this reason, dimensionality reduction is a natural precursor to analysis for high-dimensional data in different high-content screening studies [45, 27, 26].

4.1.1 Exploratory analysis using conventional approaches

Here we investigate the effects of four different methods for dimensionality reduction on a sample of our own data. This data was sub-sampled from a validation set we constructed by painstakingly finding cells which displayed one of three given phenotypes: wild-type, foci, non-round nuclei. The wild-type cells were those which displayed neither a foci phenotype, nor a non-round nucleus phenotype. The other two phenotypes are defined by either displaying a RAD52 focus (revealed in the green channel for our study), or a significantly non-round nucleus (revealed in the red channel for our study).

We compared four alternative algorithms for dimensionality reduction: PCA, Isomap [31], Local Linear Embedding [103], and kernel-PCA [44]. Other alternative algorithms such as t-SNE [33], semi-definite embedding [60], were not considered for this comparison either because they do not scale to very high dimensional spaces, or because they were similar to one of the existing comparators. For each of the four algorithms, we took a sample of 1500 data points in 916 dimensions, rescaled the data by subtracting the mean and dividing by the standard deviation in each dimension, and then produced a 2 dimensional embedding. The results are depicted in Figure 4.2.

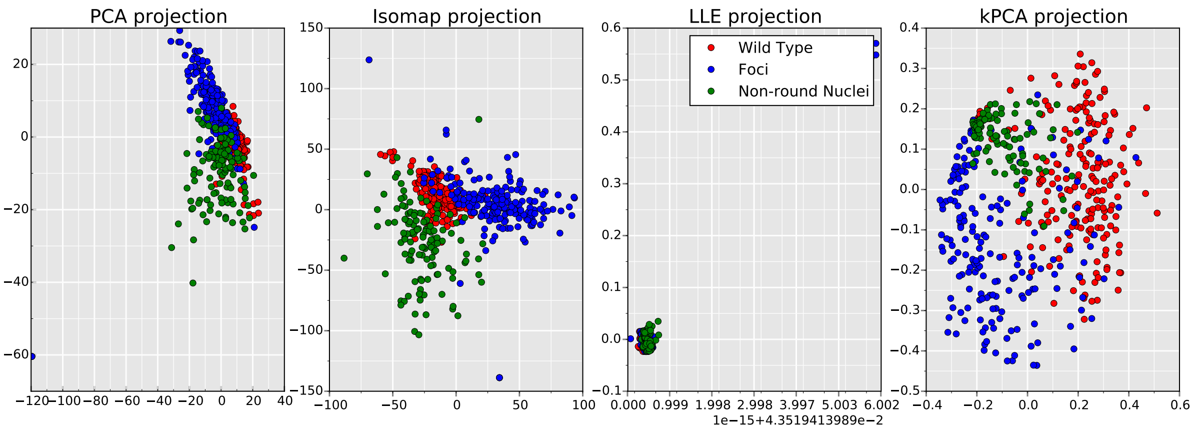


Figure 4.2: Data sampled and projected down onto a two dimensional surface. Each sub-figure shows a different dimensionality reduction method applied to the same sample of labeled data. Each cell was labeled as either a DNA damage focus phenotype (foci), a non-round nuclear phenotype (non-round nuclei), or neither of the above (wild-type). The three plots show the challenges to successfully discovering these three phenotypes for any unsupervised learning model. PCA and Isomap both force the clusters together, with significant overlap. Local Linear Embedding is less successful in preserving any of the relationships between data points. Kernel PCA shows some promise in separating the phenotypes so that they might be identified by cluster analysis.

Inspecting Figure 4.2 with respect to the 2d distribution of phenotypes gives us an intuition of what we would like in a dimensionality reduction algorithm. First, consider the contrast between PCA (the left-most panel) with Isomap (second from left). PCA maps all of these points onto one elongated cluster of points, where as Isomap is better able to separate the densely packed cluster of foci points from the non-round nuclei points. This suggests that the underlying relationship between features that represents phenotype is non-linear in nature, which a linear method such as PCA cannot effectively model. Also interesting is the contrast between Isomap and kernel PCA. While Isomap maps the points very closely together in the lower dimensional space, it also enforces a small difference between the location of the different cluster of points, likely an artifact of the many shared characteristics between each group of cells. Kernel PCA maps foci and wild-type cells into two sparse but separated clouds of points, with a cluster of non-round nuclei appearing between the two. If we wish to reduce the dimensionality in our data in a way that allows phenotypically homogeneous clusters to be recovered, then the *manner* in which we model non-linear relations between covariates matters.

Though instructive to visualize in two dimensions, we also wanted to investigate if each method would allow phenotypically homogeneous clusters to be recovered by mixture models in higher dimensions. For each comparator, we sampled points from the validation set as before, and then reduced the dimensionality of these points to a given size. We then fit a three-component Gaussian mixture model (using the EM-algorithm, with tied covariance matrices to reduce overfitting) to the reduced dimensional points, and measured the homogeneity of the final solution against the set of held-out labels from the validation set. Each run of the experiment was repeated five times for each comparator, and within these repeats, each Gaussian mixture model was fit 10 times, with the best results kept. We refer to this experiment as the *homogeneity test*, as we will revisit it later.

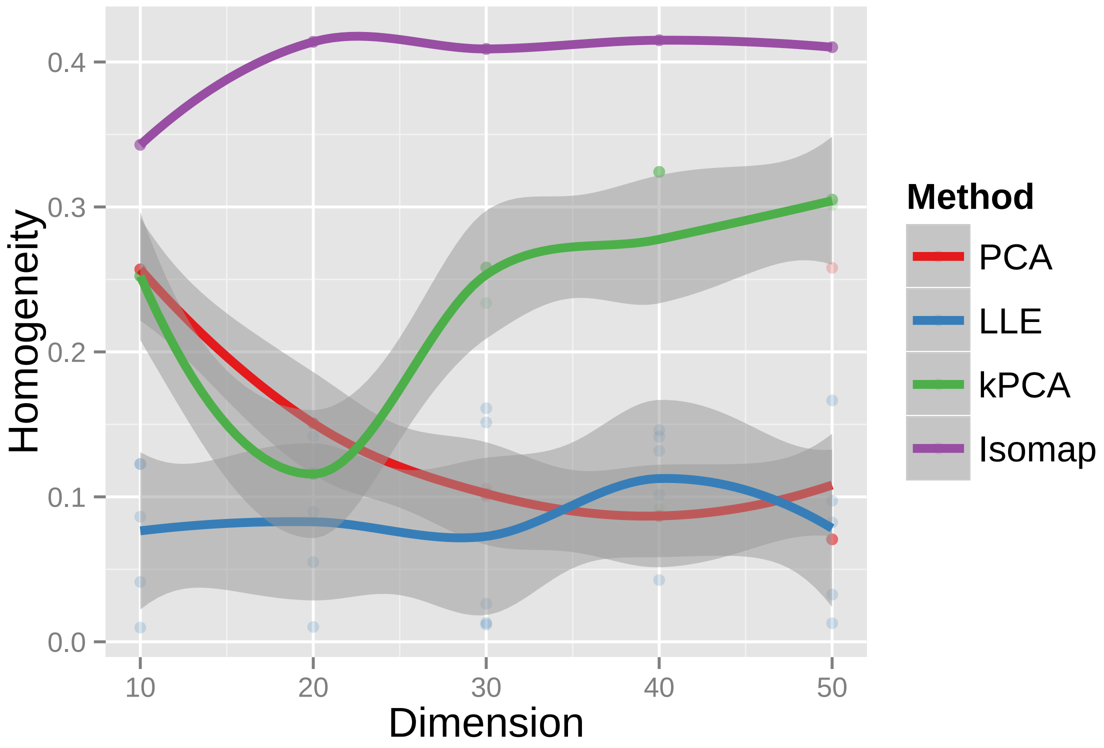


Figure 4.3: Results of the homogeneity test as described in Section 4.1.2 applied to four comparator methods: PCA, LLE, ISOMAP, and kernel PCA. Solid lines show the mean homogeneity achieved over the repeated trials, while the shadowed regions around each solid line indicate the standard deviation of values. A subset of labled data from the validation set used to measure homogeneity was reserved, and used to tune hyper-parameters for each of the methods (the number of nearest neighbours for LLE, Isomap, and the gamma parameter of the RBF kernel for kernel PCA). Hyper-parameters were found by grid-search, where the goodness of fit was measured by 5-fold cross-validated average homogeneity.

The results of the homogeneity test in Figure 4.3 gives us a clearer picture of the suitability of each algorithm as a precursor to phenotype based clustering on this dataset. They behave in very different ways. Local Linear embedding behaves poorly in across all dimensions. PCA recovers somewhat in 10 dimensions, but is otherwise poor. Kernel PCA, other than a dip in 20 dimensional space, performs steadily though not very well. Perhaps most interesting is that Isomap dominates each of the other algorithms. The consistently higher performance of Isomap, and of kernel PCA less so, in these experiments suggest that features for cellular image data do not have linear relationships, and so linear methods such as PCA will only be able to find an approximate linear transformation into a subspace that respects the shape of the data which limits their effectiveness as precursors for phenotype based clustering.

Computational or space complexity is another aspect of these algorithms which imposes a constraint on their suitability. As is alluded to in Section 2.5.1, both Isomap and Local Linear Embedding are based on an eigen-decomposition of some similarity matrix, from which each obtains a lower-dimensional embedding of the data. This has two disadvantages. The first is that the solution computed provides an embedding only for the training points, with no straightforward extension for out-of-sample examples. Bengio, Paiement and Vincent proposed extensions by reformulating both algorithms as the decomposition of a kernel matrix, and then using the Nystrom method to accelerate the process of calculating the embedding for new points [12]. However, it is not clear what degradation in performance we would observe. The second problem is that computing the original solution scales cubically in the number of data points, meaning these eigen-decomposition methods will not scale to our dataset. As [12] shows, kernel PCA suffers from a similar computational disadvantage when trying to diagonalize the kernel matrix. So, to restate what we have found in our evaluation:

- PCA is fast once the covariance matrix is formed, but is not able to model any non-linear interactions.
- Kernel PCA can model more flexible relationships, but is impractical for large datasets due to the growth of the kernel matrix. Our experiments prove that selecting the correct kernel matters greatly.
- Isomap showed impressive performance, but is impractical for large datasets.
- Local Linear Embedding did not seem to be a suitable model for our purpose.

How to best perform the dimensionality reduction remains far from clear, yet we have gained some insight. There is some evidence to support the manifold hypothesis, which posits that the data we observe actually lies on a lower-dimensional manifold, and is up-sampled to the high-dimensional space application of some non-linear functions. Therefore, we would like a method which takes advantage of this assumption, but which is not hobbled by the procedure we use to *fit* its model to large datasets. In the next section, we describe a method that satisfies both of these constraints.

4.1.2 Model evaluation via homogeneity

There are two metrics we use to evaluate models. For comparing different aspects of a model, such as when tuning hyper-parameters or comparing the effects of different activation functions, we use model reproduction error over a fixed number of epochs, averaged over either several fixture models or repeated trials. Once we have finished tuning all hyper-parameters and chosen all other aspects, we compare different models using the homogeneity test. *Homogeneity* is a measure of clustering affinity.

Given a ground truth of labels for each data point, a partition of the data induced by a clustering is homogeneous if all of its clusters contain only data points which are members of a single class [102]. A clustering of the dataset $X = (x_1, \ldots, x_N)$, each of which has a latent class drawn from $C = (c_1, \ldots, c_n)$ is a partition of X into clusters $K = (k_1, \ldots, k_m)$. Denote the correspondence between latent class labels and cluster assignments as $A = a_{ij}$, so a_{ij} is the number of data points in X with latent class label c_i also assigned to cluster k_j. The homogeneity of a clustering is a measure of the amount of information the clustering provides about the latent classes. Practically, it is defined in terms of the ratio between the conditional entropy of the classes given the clustering $H(C|K)$, and the entropy of C:

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \tag{4.1}$$

$$H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \tag{4.2}$$

Homogeneity is maximal when the clustering tells us everything about the latent class labels, i.e when the data points for each cluster are completely homogeneous with respect to their latent class labels. In this case, $H(C|K) = 0$. By convention that 1 is good, 0 is bad, homogeneity is calculated as:

$$h = 1 - \frac{H(C|K)}{H(C)} \tag{4.3}$$

Since labels in a clustering result are only meaningful to indicate differences between clusters, homogeneity is independent of the label values, and is invariant under permutation of the class or cluster indicator values. For example, if we cluster the data with latent labels $(1, 2, 2, 2, 3)$ as $\{1, 2, 2\}\,\{2, 3\}$, this is just as homogeneous a solution as $\{2, 2, 3\}\,\{1, 2\}$. We propose to use this metric to evaluate the goodness of a dimensionality reduction algorithm as follows:

1. reduce data in the validation set.
2. fit a Gaussian mixture model to the reduced data, pick the best fit over several repeated trials to account for initialization.
3. measure and record the model homogeneity.
4. repeat the previous three steps five times, and report the mean and standard deviation of homogeneity.

Given that we would like to find clusters in the lower dimensional data that reflect different phenotypes, we reason that this test will give us the best chance of discovering which dimensionality algorithm will provide the optimal lower-dimensional encoding of the data for a clustering model to find a solution whose clusters are phenotypically homogeneous.

4.2 Dimensionality reduction with autoencoders

Dimensionality reduction can stated as a search over functions. Given a high-dimensional dataset $X := X_1, \ldots, X_n$, produce an injective mapping $f : X \to Y$ from the high-dimensional data space X to the low-dimensional latent space Y based on some desired properties imposed on the resulting $Y_1, \ldots, Y_n$. For example, one formulation of PCA parameterizes f as an orthogonal projection which maximizes the variance among the projected data $Y_1, \ldots, Y_n$. The idea of using a neural network to parameterize the function f first considered in the late 1980s [7]. Baldi and Hornik proved that a neural network with linear activation functions on its units would produce an optimal solution that spanned the same space as the principal components of the covariance matrix of the original data. In other words, its performance measured by information loss was dominated by PCA, but without the structure provided by principal components (i.e orthogonality). A decade later, work by Japkowicz et al. showed that if the hidden units of the neural network had non-linear activation functions, then a neural network acting as an autoassociator (a synonym for autoencoder) had the power to find very different solutions than PCA, and provided a promising new approach to finding non-linear structure in high-dimensional data [55]. However, this occurred at about the time when neural networks were losing favour in the machine learning community, and so there was little progress on using them for dimensionality reduction [1]. Interest was renewed when in 2006, Hinton and Salakhutdinov published a seminal paper describing how to perform dimensionality reduction [50] using deep autoencoders. They constructed a multilayer neural network which learned to compute low-dimensional codes for high-dimensional data, that could be trained unsupervised, and which showed promising results on several benchmark datasets. The chief innovation that led to the success of this work was a novel way of initializing the network weights in a layer-wise manner, so that the error between the original data and the network reconstruction was low. This scheme was referred to as unsupervised *pre-training*.

Pre-training was introduced as a technique for initializing neural networks layer by layer in a greedy fashion [51, 11, 96]. Each layer is first trained to produce a latent representation of the observed data, by minimizing the error between the observed and reconstructed data. Once one layer has been trained, the subsequent layer receives as input the latent representation of the previous layer, and the procedure is repeated to set its parameters. The result is a model which transforms the input patterns into an output that lies at a different level of abstraction. This process is repeated as many times as desired, which results in a hierarchical composition of several models, hence the name 'deep' network. Once the pre-training stage is completed, the parameters of the network are *fine-tuned*: all the parameters are updated in parallel, according to the optimization of some higher level objective. For networks intended for classification, the top-most layer will provide its output to a further layer that performs classification, such as a logistic regression or linear SVM, and the higher level objective becomes the minimization of negative log-likelihood. For dimensionality reduction, the higher level objective is the minimization of the error between the original data and the model reconstruction [50, 105].

Hinton and Salakhutdinov used probabilistic models called Restricted Boltzmann Machines as their base model for pre-training each layer of the larger model. A similar line of work developed by Bengio, Vincent, Larochelle and Erhan used denoising autoencoders as a basis for pre-training deep autoencoder models. We chose to use denoising autoencoders as the base model for our own experiments. Restricted Boltzman Machines are probabilistic models over visible and hidden units. To learn the parameters of the model (such as edge weights connecting visible and hidden units) by maximizing the likelihood of the data given the model parameters, you must calculate the partition function, which is a sum over all possible configuration of visible and hidden units. This is intractable, and so must be estimated. This

[1] I do not want to digress too far into the recent history of machine learning, but the reasons for this decline in interest can be attributed to a difficulty with training networks with multiple layers of hidden units, no real idea of how to initialize networks properly, and the simultaneous development of support vector machines

can cause instability during training. De-noising autoencoders optimize a tractable objective, and so it is easier to interpret and correct implementation errors.

4.2.1 Stacked denoising autoencoders

Recall from Section 2.5.1 the definition of a denoising autoencoder, which must reproduce a clean version of its input, given only a noisy version of that vector. This is accomplished by learning a function that maps vectors from the input space into a lower dimensional space, as well as the inverse of that function. When composed with each other, denoising autoencoders have been used successfully to build complex functions mapping high dimensional data into low dimensional spaces. Previous work on stacking autoencoder models by Bengio, Larochelle et al. [11, 62], as well as the proven utility of denoising regularization for unsupervised pre-training [124] lead to the stacked denoising autoencoder model [124, 125]. In a manner analogous to how Hinton and Salakhutdinov used neural networks, we propose to use a stacked denoising autoencoder (SdA) as a parametric model for dimensionality reduction. In the formulation proposed earlier, $f_{sda} : X \rightarrow Y$ is parameterized by a stack of n denoising autoencoders $A_0, \ldots, A_n$, each with parameters $W_0, \ldots, W_n$ (recall the definition of the dA in 2.5.1).

Stacked denoising autoencoders are well suited to reducing data from high-content screening data. They have been shown to perform on par with deep neural network models in many object recognition tasks [125, 124]. They are not hampered by the out-of-sample issues that plague many manifold learning techniques, yet they are expressive enough to learn complex non-linear manifolds. They can scale to

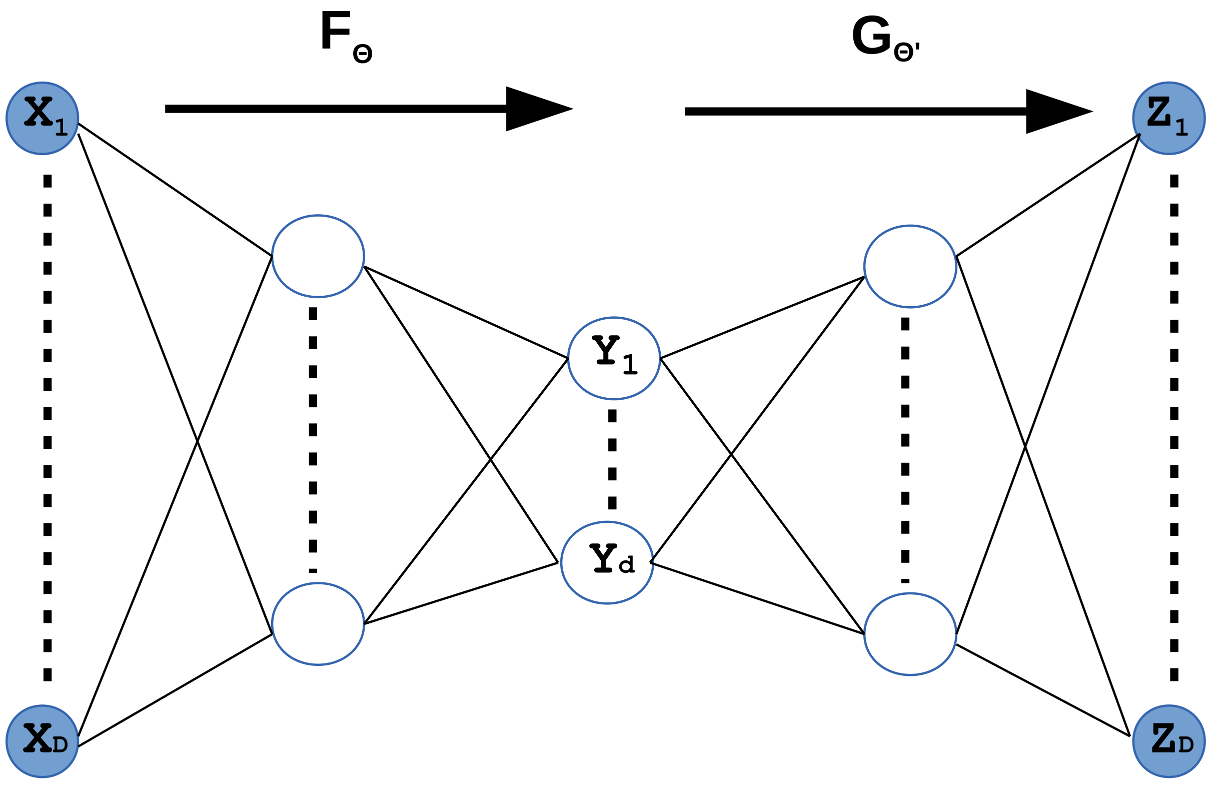


Figure 4.4: A diagram of how the unrolled stacked denoising autoencoder goes from input data, up through the different layers of representation, and back through the inverse transformation to reconstructed input.

any sized dataset at test time. Their training time can be reduced by accelerating the computation on GPUs, which enable us to explore many models and train on large input data.

4.2.2 Learning

We learn the parameters of the SdA by minimizing the loss between the training data and its reconstruction under the model. Suppose that our SdA model is composed of k denoising autoencoders, each with encoding and decoding functions f_k, g_k. For real valued data, the reconstruction error over a batch of examples $X_1, \ldots, X_n \quad | X_i \in \Re^D$ is measured as

$$L(X, Z) = \sum_{batches} \sum_{i=1}^{n} (X_i - Z_i)^2 \tag{4.4}$$

$$Z_i := g_1 (\ldots \ (g_{k-1} (g_k (f_k (\ldots \ (f_2 (f_1(X_i)))))))) \tag{4.5}$$

$$g_j = g_{\theta'_j}(y) \ = \ s(W'_j y + b'_j) \ j \in [1 \ldots k] \tag{4.6}$$

$$f_j = f_{\theta_j}(x) \ = \ s(W_j x + b_j) \ j \in [1 \ldots k] \tag{4.7}$$

that is, the squared error between each vector X_i with its corresponding reconstruction Z_i. The parameters upon which $L(X, Z)$ depend are the set of transformation matrices W_j, and bias vectors b_j for encoding and decoding functions f_j, g_j respectively. Since the dataset is large, we break it down into mini batches, and perform stochastic gradient descent with minibatches to minimize L with respect to the parameters of $f_j, g_j \forall j$. Our dataset fits into memory, so to compute L, we repeat the following steps: sample randomly from one of the batches, compute its updates to the parameters, and apply the updates. This is performed until we cease to improve our reconstruction error on a validation set.

Pre-processing

We did little pre-processing of the data except to filter out improbable cells, that can occur by virtue of segmentation errors. For a large sample of the labeled validation data, we computed density plots over each of the features with geometric intuition. We then discarded any objects that exceeded two standard deviations either side of the mean (cf. Figure 4.6). For those cells which remained, we standardized them by subtracting the mean and dividing by the standard deviation over each feature.

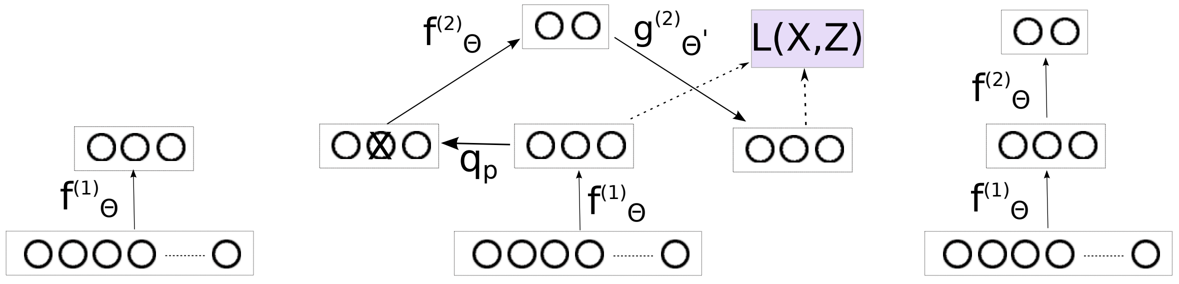


Figure 4.5: Schematic representation of an SdA. The model is a product of layering dA models such that the hidden layer of one model is treated as the input layer of the next dA. The hidden layer of the top-most dA represents the reduced data.

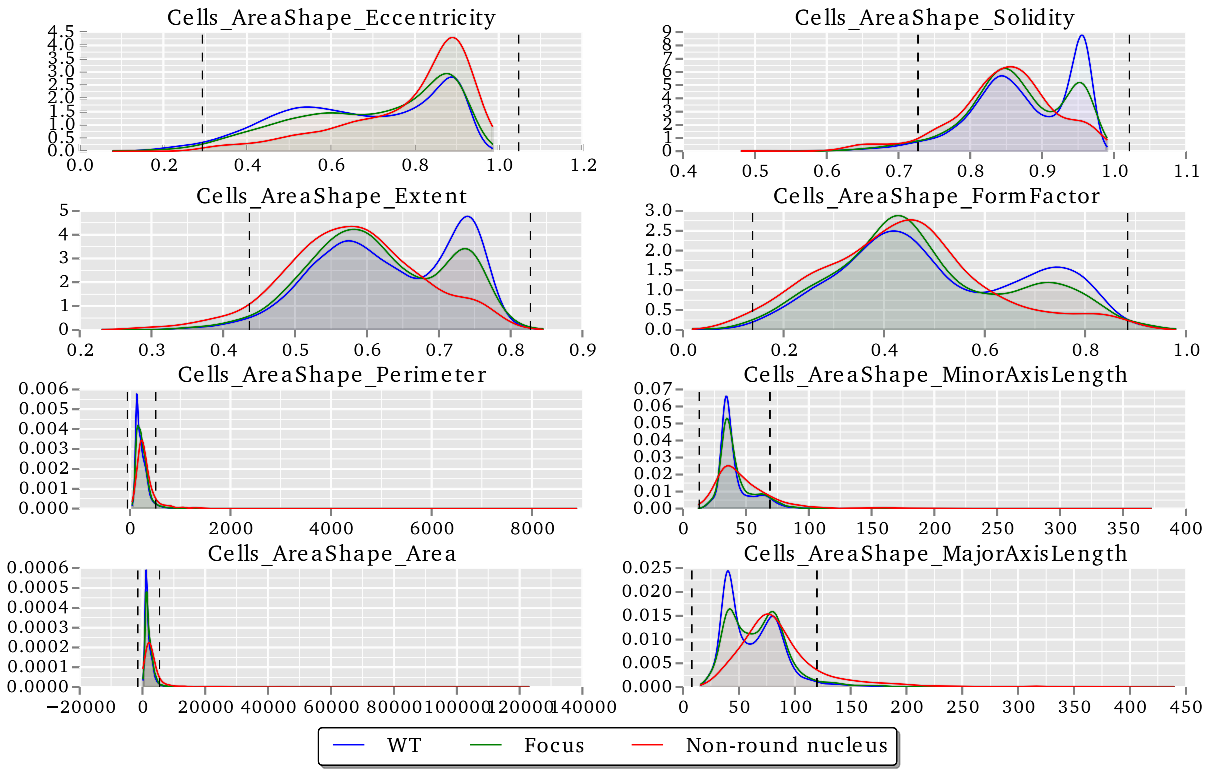


Figure 4.6: Density plots over geometrically interpretable cell features. Each coloured line is the smoothed kernel density estimate for a sample of wild-type (WT), DNA damage foci (Focus) and non-round nuclear (Non-round nucleus) labeled cells drawn from the validation set. Dashed vertical lines represent two standard deviations from the mean for wild-type cells, and were used to

Pre-training

Each layer in each SdA was pre-trained by performing stochastic gradient descent on mini-batches for 50 epochs, updating the parameters for each encoding and decoding function: $W_{vis}, W_{hidden}, b_{vis}, b_{hidden}$. For the first layer we used a sigmoid activation function and a linear decoding function, so that we reproduce real valued data. For every dA higher on the stack, both encoding and decoding functions w and the decoding function are sigmoid. Each layer was trained to minimize the mean squared reconstruction error of the input, averaged over the mini-batch. This meant higher layers used Bernoulli valued layers, so we used cross-entropy as the loss. We describe our search over hyper-parameters below.

Fine-tuning

Once pre-training for all model architectures was completed, each model was fine-tuned. The network is 'un-rolled', meaning that data is input to the bottom layer, propagated up through the network to the highest (low-dimensional) layer, and this output is then decoded back down the network layers with the final output being a reconstruction of the original input. Fine-tuning for these models consisted of performing 200 epochs of stochastic gradient descent in mini-batches of size 500 on a set of approximately 650,000 sampled data points. We also used a validation set of approximately 10,000 data points. An extensive search over hyper-parameters and model capacity was performed, described below. Reconstruction error on the validation set was measured at the end of each training epoch. Training was stopped when the number of epochs was exhausted, or improvement on the validation set dropped below a 0.5% improvement on the best measured performance.

Hyper-parameter tuning

We performed a grid based search over a variety of hyper-parameters in our model. When training models with large numbers of parameters using stochastic gradient descent, there you must examine how performance varies with respect to a number of hyper parameters including the learning rate, the momentum, the regularization parameters, the mini batch size, and in our case the amount of injected noise. We present the results of our experiments in Figure 4.7.

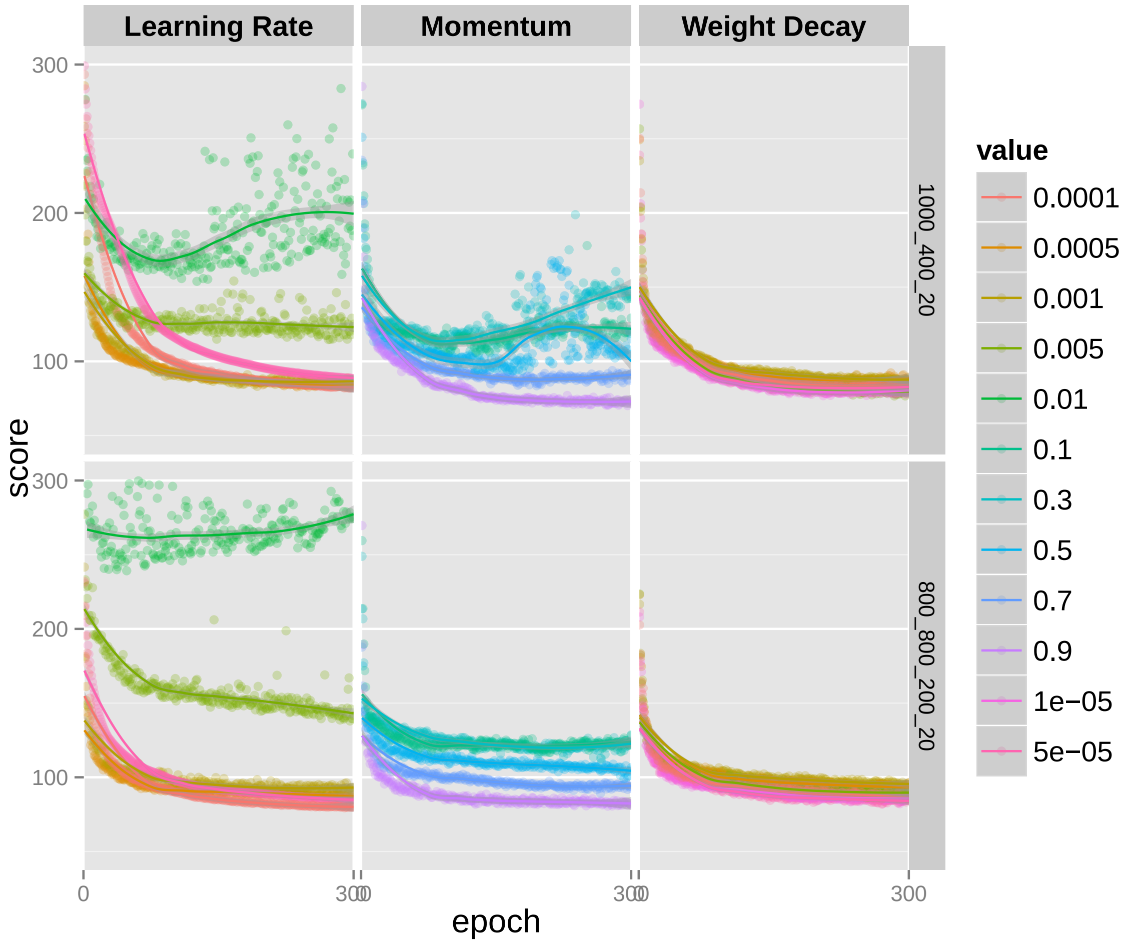


Figure 4.7: Reconstruction error of two different SdA models, on a grid of hyperparameter points. Left panels, top and bottom: reproduction error for different starting values of the master learning rate. Middle panels, top and bottom:

We restricted the tuning of our hyper-parameters to a set of fixture models (identified by their layer sizes in the right hand margins of Figure 4.7), to try and limit the amount of computation required. In each test, we used the reconstruction error over a fixed number of epochs over a previously selected random subset of our data to determine the optimal value. The parameters we tuned were the following:

Learning rate: the rate modifying the magnitude of the gradient update to model weights.

Noise rate: the proportion of the input units in each layer subjected to masking noise during pre-training of each dA layer.

Momentum: the fraction of the previous parameter update value carried over to contribute to the current update value.

Weight decay: the regularization penalty on the dA weight matrix parameter updates.

Results showed that the value of learning rate had a noticeable effect on the ability of a layer to reconstruct the input data, while the other parameters showed little variation.

Stochastic gradient descent

Fitting a large number of model parameters using stochastic gradient descent on a strongly non-convex objective can result in mixed performance. There are different variations that each have some formulation to try and increase the chances of converging on a sub-optimal solution. We experimented with different variations of stochastic gradient descent, using the reconstruction error test as with our hyper-parameter search. We compared stochasitc gradient descent using classical momentum, Nesterov's accelerated gradient descent, Adagrad [35], and a variant of Adagrad with a sliding window of past gradient evaluations called Adadelta [133]. We used two fixture models for each model depth, and ran a bank of five trials. We report the mean reconstruction error and standard deviation (shaded band).

Figure 4.8 show that both Adagrad variants outperform the other methods. Results for Nesterov's accelerated gradient are not shown, as they were a factor of three higher than the others. We chose to use Adagrad WD, since though it was virtually indistinguishable from Adagrad, it has the possibility to temporarily increase individual components of the learning rate vector should that given component would be less sensitive to the eventual vanishing of Adagrad's adjusted learning rate.

4.2.3 Architectures, layer sizes and depth

We experimented with a variety of layer sizes, number of units per layer, and differnt types of activation functions in our models. Since we used a grid over layer sizes, we were limited in the depth at which we could fit and evaluate layers. While we could run an effective grid search over 3 layer and 4 layer models in a matter of approximately 24 hours of computing time (that is pre-train and fine-tune all models with 2, 3 hidden layers respectively), the same tasks for five layer models was computationally prohibitive. In any case, we did not notice a vast difference in the reduction of reconstruction error for the 5 layer models we did fit.

Layer size and depth

To test the effect of layers and layer sizes, we set up a search over a grid of unit capacities for each of three and four layers. We chose to begin with a limited four layer architecture was based on previous work by Hinton and Salakhutdinov [50]. Each model in the search was a point on the following grid, where each layer size set had a step size of 100: $\{700, \ldots, 1500\} \times \{500, \ldots, 900\} \times \{100, \ldots, 400\} \times \{50, \ldots, 10\}$. The layer size model search was run independently of the hyper-parameter model search. We performed

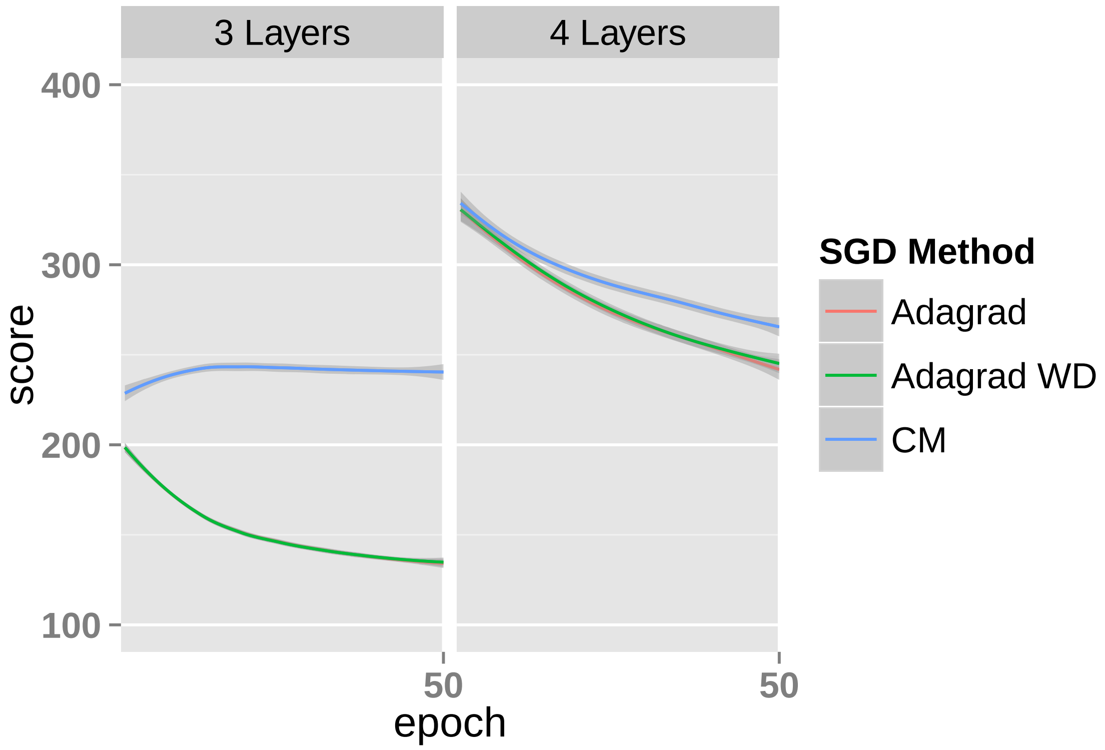


Figure 4.8: Reconstruction error for models trained using stochastic gradient descent for a fixed number of epochs, regularized with classical momentum, Adagrad, and Adadelta. In both panels, Adagrad and Adagrad WD (red, green) outperform stochastic gradient descent with classical momentum (blue).

our architecture search in parallel on a large GPU computing cluster. Each node in the cluster had two GPU cards, and the model architecture search grid was set up to ensure an even number of jobs to allow for two model architectures per node. All GPU code was written in Theano, a mathematical expression compiler written in Python [13].

Activation functions

The choice of non-linear activation function helps empower autoencoders to model more complex functions. Activation functions should be readily differentiable and fast to compute. Two candidates we considered were sigmoid activation functions ($f(x) = \frac{1}{1+e^{-x}}$) and rectified linear activations ($f(x) = max(0, x)$) (abbrv. ReLU). For all layers with ReLU activation, we used both linear encoder and decoders. We ran an experiment to compare the two unit types. For two given SdA models, we fixed all other parameters save for activation, and trained them using stochastic gradient descent with momentum for 50 epochs of training. We recorded the average reconstruction error scores shown in Figure 4.9.

For our task, the sigmoid activation function consistently provided better empirical performance in reconstruction. We experienced great difficulty getting the ReLU units to learn, and had to resort to several tricks for controlling extreme behavior such as gradient clipping. We hypothesize that the hard cutoff of the ReLU unit introduces errors at each layer of the SdA during reconstruction. Pre-training of each layer depends on the frozen parameters of the previous layers. If at any layer there is a zero

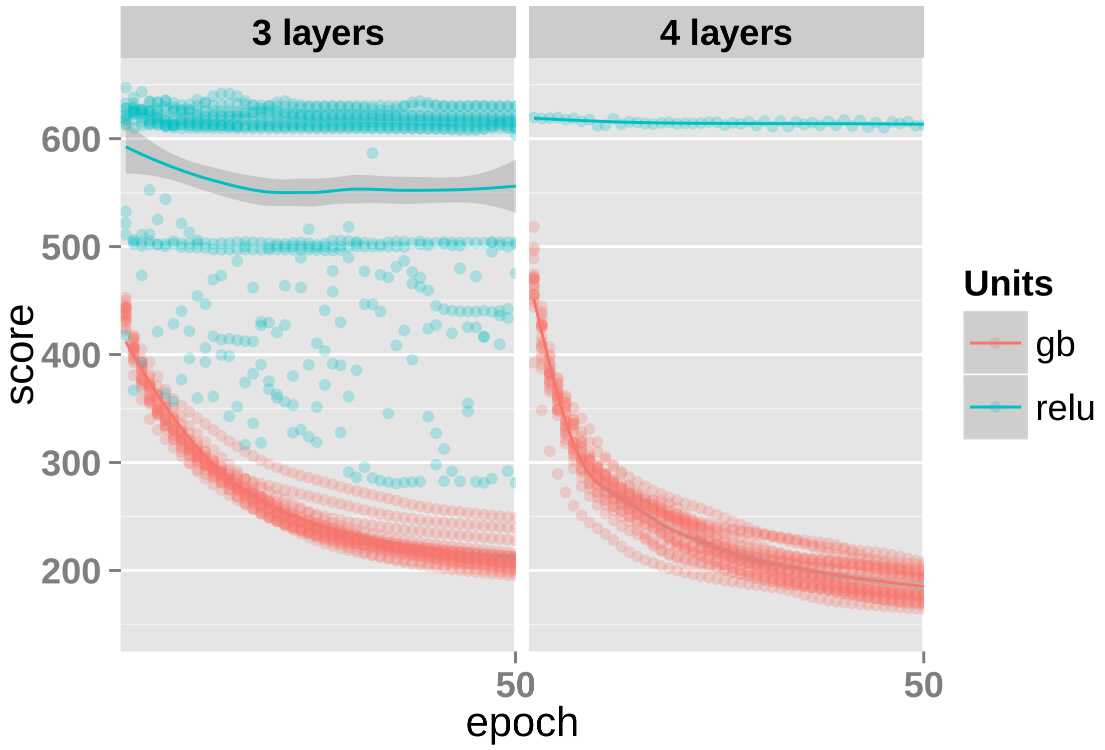


Figure 4.9: Comparing activation functions: (sigmoid/linear) versus (ReLU/linear). Score refers to the average reconstruction error.

appears, then local training of that layer will reproduce that zero perfectly. However, this will limit the model's ability to do learning in fine-tuning, since the local layers have over-fit zeros in their local pre-training. We observed several pre-training runs where the first layer of the SdA achieved modest reconstruction error, the second layer quickly dropped to nearly zero reconstruction error, and yet the model would have astronomical fine-tuning reconstruction error. We conclude that ReLU units do not work well in layer-wise pre-training, as was confirmed by Glorot et al. [42].

4.3 Results

To compare our models against the other manifold learning algorithms as well as PCA, we performed several rounds of the homogeneity test using a validation set labeled cells which was generated by first manually labeling a collection of cell objects representing each phenotype, and then training an SVM in a one versus all manner to label the remaining cells. The SVM predicted labels were manually validated. We pre-processed the data in the same manner as for training. The scaled data was then reduced using one of the candidate algorithms, followed by Gaussian mixture clustering using scikit-learn [1]. The number of mixture components was chosen as the number of labels in the validation set. We report the average homogeneity (fit by loess) as well as the standard deviation 4.10 for the top 5 performing runs.

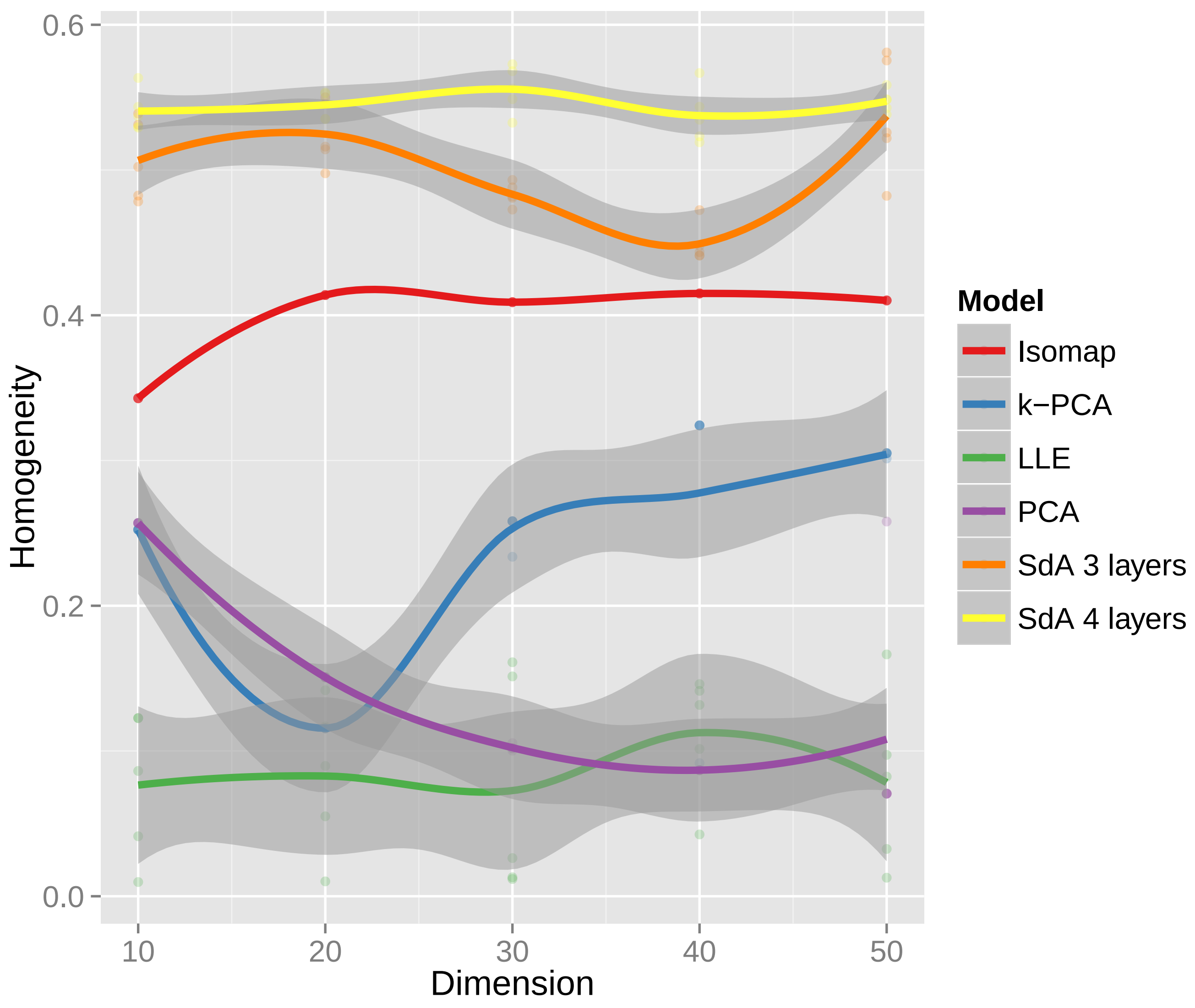
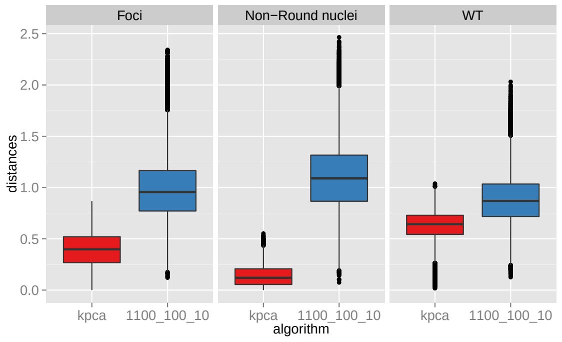


Figure 4.10: Homogeneity test results of 3, 4 layer SdA models versus comparators. For each model, the data was reduced from 916 dimensions down to 50..10 dimensions. For each model and embedding of the data, a Gaussian Mixture Model was run with randomly initialized parameters for 10 iterations. Each of these runs was repeated 10 times. The parameters from the best performing run were used to initialize a GMM which was run until convergence, and the homogeneity was measured and recorded. This process was repeated five times for each model, resulting in five homogeneity measurements. We report the average homogeneity (solid line) as well as the standard deviation (shadowed gray).

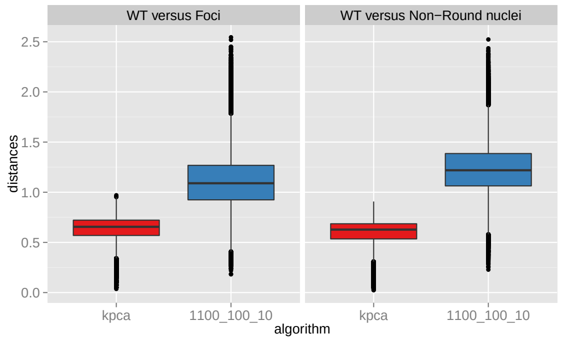
4.4 Discussion

In this chapter, we have presented and explored the use of deep autoencoders for dimensionality reduction applied to high content screening data. Building the model layer by layer with denoising autoencoders, and then minimizing a global reconstruction objective via mini-batch stochastic gradient descent allowed us to achieve good performance relative to other comparable methods. Examination of Figure 4.10 shows that SdA models were consistently ranked as the best models for dimensionality reduction in preparation for phenotype based clustering. Interestingly, there was no appreciable loss in homogeneity for SdA models as the dimensionality of the space shrank, as was observed for the best-performing comparator Isomap. Also interesting is that while the best performing 4 layer models dominated the best performing 3 layer models, the results were not significantly greater in the 10 and 20 dimensional spaces where we would prefer to operate. This is not unexpected; an SdA model with only 2 hidden layers has less modeling capacity than one with 3 hidden layers, so it is not so surprising that the higher capacity models will find a better approximation to manifold. The closeness in performance also suggests we will not see much gain in performance by using even higher capacity 5 layer models.

To try and understand what accounts for the gap between SdA models and the other algorithms, we randomly sampled output from each algorithm and computed estimates of the distribution over both the inter-label and intra-label distances between points. Shown in Figure 4.11 is an example of the estimated distance distributions between kernel PCA and a sample three layer SdA model. While the points sampled from kernel PCA have a smaller intra-label mean distance[2], the points sampled from the SdA model have a larger inter-label distance[3]. Therefore, a clustering algorithm applied to SdA reduced data was more reliably able to find assignments that reflected the phenotype labels.



(a) Within-class distance distributions, for SdA (denoted by its architecture) versus kernel PCA



(b) Distance distributions from points sampled from different classes, for SdA versus kernel PCA

Figure 4.11: Sampling from distributions over distances of data points reduced to 10 dimensions by both SdA, and kernel PCA. While the SdA models do not always produce the tightest packing of points within classes (see left), they do consistently assign the widest mean distances to inter-class points.

Finally, to perform a visual inspection of the results of our models, we repeated the 2 dimensional reduction experiment whose results are depicted in Figure 4.2 with our SdA models. We see that in comparison to the models in Figure 4.2, SdA models are able to provide better separation of points with different phenotypes, offering one explanation for their superior performance on the homogeneity test.

To summarize, we have introduced deep autoencoder models for dimensionality reduction of high content screening data. Mini-batch stochastic gradient descent allowed us to train using hundreds of thousands of data points, and the nature of the model allowed us to apply the resulting models to unseen data, circumventing the limitations of other comparable dimensionality reduction algorithms. We also demonstrated that SdA models produced output that was more easily assigned to clusters that reflected biologically meaningful phenotypes.

[2]Averaged over all three class labels
[3]Only one comparison is shown for conciseness, but this pattern is consistent across the other algorithms

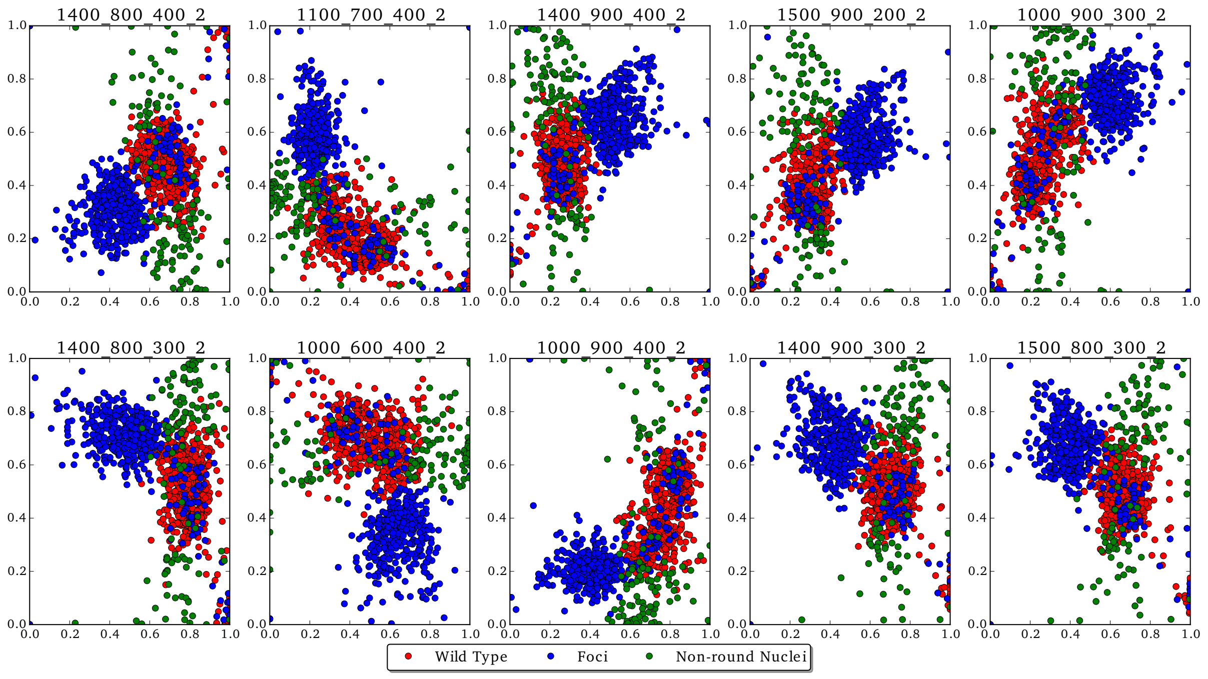


Figure 4.12: Ten randomly chosen models with output layer of size 2 for visualization purposes, fit to a subset of the labeled validation data.

Chapter 5

Characterizing populations via non-parametric Bayesian clustering

In the previous chapter, we developed a method for performing dimensionality reduction on HCS data. The method dealt with large N $\gg$ P data for which PCA is unsuitable due to non-linear relations between covariates. We used it to build a deep autoencoder model that allowed us to reduce the data to a smaller dimensional space where modeling it becomes more tractable for unsupervised learning.

When the cost of gathering labeled data becomes too high, supervised learning ceases to be a suitable framework for analyzing HCS data. If we can no longer rely on labeled data for training classifiers to predict the phenotypes of unlabeled cells, then we must consider unsupervised methods. In Chapter three, our focus was on discovering genes whose populations had highly skewed proportions of cells that displayed DNA damage foci when compared to wild-type populations (cf. Section 3.4.4). Here, we develop an unsupervised method with the same goal: finding populations that are poorly represented with respect to wild type. The difference is that without labels, we do not have an a priori definition of the phenotypes we expect to encounter in each population. Instead, we want to build a density model using wild-type data and to use the model to find populations whose own densities differ from the wild-type model.

This is naturally formalized as an outlier detection problem: given a mixture model of wild-type data, find populations whose data does not fit the wild-type density model. Anomalous populations may be rare, which would makes them difficult to find. Thus our goal becomes identifying an appropriate density model for amalgamated wild-type data and identifying those genes whose cell populations are not well fit by the wild-type density model.

In this chapter, we describe a three stage method for finding mutant populations that display different proportions of phenotypes relative to a control (or wild-type) population. The first stage is to reduce the dimensionality of the data. The next stage is to fit a nonparametric mixture model to a randomly sampled set of the reduced data, to establish a reference density for the data set. The final stage is to evaluate the posterior mixture proportions of each individual mutant population under the reference model parameters and to compare the observed posterior mixing proportions with respect to the reference population to detect interesting outliers.

5.1 Clustering lower dimensional data high-content screening data

In the HCS literature, cells in a population (subjected to some treatment) are commonly represented as a collection of features and the variability of the features among different populations are modeled by some supervised learning algorithm [73, 6, 38]. The responses of each population under the model are then either ranked or clustered, and the resultant list or clusters are then mined for biological insights. While this design has been successful, it depends on a great amount of human annotation, which limits how much biological knowledge can be distilled from the data analysis. Recent work by several groups has used unsupervised learning to analyze high-content screening data [131, 112, 98]. Unsupervised methods are usually motivated by a desire to reduce the amount of manual labour, or to avoid preconceived biases in the data analysis. These methods tend to employ a recipe of reducing the dimensionality of the initial data, either by feature selection or by PCA, and then fitting a Gaussian mixture model (or set of models) to the reduced data.

5.1.1 Exposition of related work

To motivate our method, we review two recent papers that implement a variation of this recipe to characterize populations of cells in high content screening data as mixtures of different phenotypes.

Iterative cluster merging for online phenotype discovery Yin et al. ([131]) present an algorithm for online learning of phenotypes, represented by clusters of cells. Their method analyzes each image sequentially while scanning progressively through a large image data set. Their goal is to identify all the different biologically meaningful phenotypes present in the data set. They begin with an unlabeled training set of cells, represented as a bag of features. They then discard all but 15 features using a method that performs K-nearest neighbours on the set of features based on a covariance similarity measure [79]. They partition that training set into different phenotypes K_0, each of which is represented by a Gaussian mixture model. The algorithm then proceeds iteratively through each image, trying to match the cells in that image to one of the identified phenotypes. If it cannot do so, then the un-matched cells are assigned to a new phenotype also represented by a Gaussian mixture model. The algorithm proceeds as follows, iterating through phenotype models S_m and images:

- Cells from the image E are combined with a sample drawn from the current phenotype S_m into a new set F.
- Based on the distribution of the data in F, the optimal gap statistic c is computed, which is an estimate of the number of clusters under which intra-cluster distance is minimized and inter-cluster distance is maximized [119]. Clustering with c exemplars is performed on F.
- For each of the identified c clusters, the composition in terms of points from E and S_m is examined. If any clusters have fewer than 5% of points from the image E versus from the phenotype S_m, then those points are merged with the phenotype S_m. Otherwise, the points remain in E, and the next phenotype is considered.

This process is repeated for each known phenotype. If any clusters of points drawn from E remain after each phenotype S_m is processed, then each of these forms the basis of a new phenotype. A Gaussian mixture model is fit to each new phenotype, and added to the K_0 other phenotypes. Yin et al. evaluated their algorithm on a synthetic shapes dataset and on two biological datasets (Drosophila embryo images and stained HeLa cells).

Characterizing heterogeneous cellular responses to perturbations Slack et al. [112] presented a method for characterizing the variation of cellular phenotypes in HeLa cells treated with different drugs. Each population of HeLa cells was subjected to a different dosage of a given drug, and each cell was represented as a collection of pixel based features subsequently reduced using PCA to the first 25 principal components. They then construct a reference model by sampling 10,000 cells[1] randomly from each population in the study, and using this amalgamated data to fit a Gaussian mixture model.

They defined heterogeneity of a treatment population as the sum of the posterior distribution of mixing proportions for all cells evaluated under the reference model, after re-normalizing. Profiles for each drug treatment condition were computed by an average of profiles for each replicate well, then averaged by the number of cells observed in each well. They compared the similarity among distribution profiles by a symmetric version of the KL-divergence:

$$S(P,Q) = \frac{KL(Q,P) + KL(P,Q)}{2}$$

and noted that populations subjected to drugs with similar modes of action frequently clustered together.

5.1.2 Proposed methodological improvements

These highlighted methods suffer from methodological limitations. First, PCA may produce representations that not are amenable to clustering when clusters are meant to represent phenotypes, as shown in the previous chapter. Second, selecting the number of mixture components a priori is difficult. Estimation by cross-validation requires that the fitting procedure is run to convergence many times, with many different initializations of the parameters, and that the average likelihood of the model for each value of K is compared. This quickly becomes a computationally intensive process. Even then, we must be careful that we do not encounter singularities in components due to outliers when we fit by maximum-likelihood. Heuristic methods to estimate the number of components, such as the so-called "elbow graph" of explained variance (or marginal likelihood) using bootstrap methods [119], may be intuitively appealing but offers no certainty. Both the cross-validation and bootstrap methods are extremely computationally intensive and do not scale well to very large datasets.

An appealing alternative is to employ nonparametric models (see 2.5.3). More flexible and expressive ways of constructing mixture models have existed in the machine learning literature for approximately 15 years (cf. [97]) and in the statistics literature well before then. The appeal of these models, especially from a Bayesian viewpoint, is that they help combat overfitting and adaptively select for model capacity as part of the usual modeling procedure. We proposed to improve upon the approach of Slack et al. for profiling populations in HCS by updating the tools with which these profiles are constructed. Below, we describe our approach for constructing a reference model using non-parametric Bayesian clustering. We applied our model to characterize the populations in the single deletion mutants dataset as described in Chapter three and compare the results to those of our supervised learning algorithm.

[1]roughly 1 per-cent of all cells in the study

5.2 Outlier detection by nonparametric Bayesian clustering

We proposed to use a DPMM to estimate our reference population in place of the standard Gaussian mixture. The basis of DPMM was introduced in Section 2.5.3. Mixture models can be split into a model to allocate points to clusters and a model that generates these observed data points given the cluster allocation. We used the Chinese Restaurant Process formulation of the DPMM for the allocation model. Our observation model was Gaussian with unknown mean and inverse covariance (or precision). We used normal-inverse Wishart conjugate priors for the parameters of each component.

5.2.1 Sampling from the populations to form the reference data set

We constructed a reference data set for the single deletion screen condition by first reducing the entire data set. The reduced data is the low-dimensional encoded output from an SdA model. Fewer dimensions is preferable when clustering, and since we did not observe any loss in performance as measured by the homogeneity test (cf. Figure 4.10), we chose to reduce the data to 10 dimensions. Given the close overlap in performance on the homogeneity tests for top performing three and four layer SdA models, we created two reduced data sets, using both a model from the three layer set $\{1500, 300, 10\}$ and from the four layer set $\{1400, 700, 400, 10\}$. The homogeneity test scores for these two models were 0.538 versus 0.536.

We sampled 5 percent of cells, uniformly at random without replacement, from each well, which amounts to a population of 28,876 cells. In contrast, Slack et al. sampled approximately 1 percent of cells from their population, a ratio we felt would be far too small for fitting a reference model given the variability in sampling observed in Figure 3.6. We tried to sample larger proportions of cells, but the increased volume of data lead to lengthy computation times for our MCMC sampler.

5.2.2 Fitting model parameters

We investigated two different methods to fit the DPMM model parameters. Both are described in detail in Section 2.5.4. The first method is a collapsed Gibbs sampler where the state of the Markov chain consists solely of the indicators that represent the association of data point to mixture component. This method was first presented by MacEachern and is described as "Algorithm 3"[2] in an excellent review of MCMC methods for DPMMs by Neal [85]. The second method is a more recent variational inference method developed with DPMMs in mind, which scales well to large data sets [53].

5.2.3 Experiments for constructing reference population model

In constructing a reference model, we were interested not only in the posterior of the number of active components, but also in which dataset to use. We ran repeated experiments to fit DPMMs using both inference methods, and examined the posterior distribution of the number of active components for the aggregated results. In addition, we contrasted the two data reduction models by comparing the lower bound on the evidence for DPMM models fit by variational inference.

Collapsed Gibbs sampling experiments There is a trade to be made when fitting MCMC models between sampling for a longer period versus pursuing more chains. Should we opt for running more independent chains, or should we spend our time running fewer chains but for a longer period? Though

[2]almost as if an extra in a film

collapsing the number of latent variables should increase sampling efficiency in theory, each sampling iteration, or complete pass through the data set, took approximately 91s of wall-clock time. As a compromise, we ran 10 samplers in parallel for each of the two reduced data sets. We considered a burn-in period of 50 sampling iterations, and then ran for a further 800 iterations.

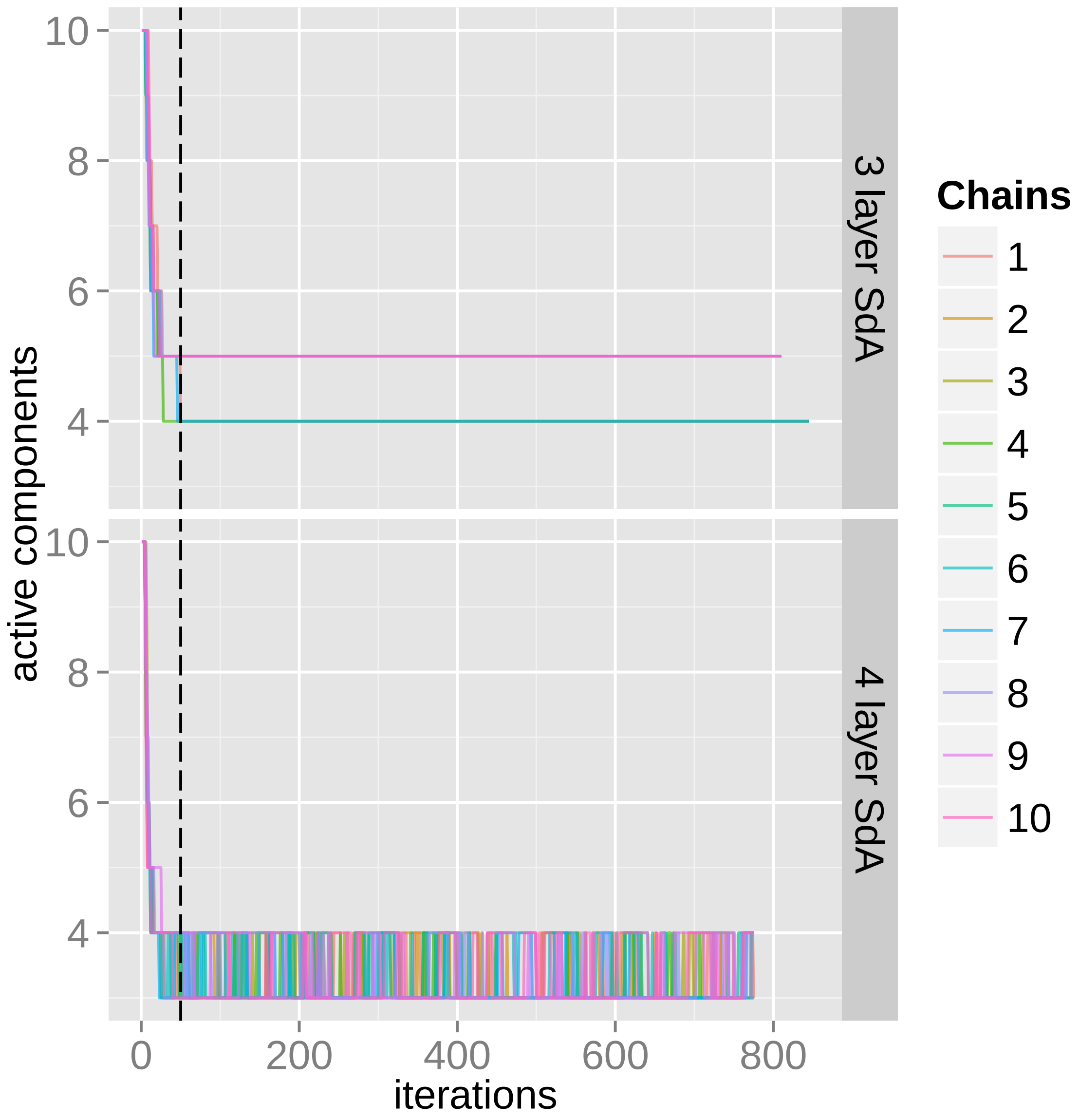


Figure 5.1: Trace plots of the Gibbs samplers for number of active components, for chains run with the 3 layer data (top) as well as the 4 layer data (bottom). The dashed dark line indicates the 50 iteration mark.

The results show that chains run on the three layer data are split between supporting four or five components, whereas the chains run on the four layer data are split between three and four components. All the chains in both experiments move strongly towards one of the two modes they favour, suggesting that we should favour models with fewer components.

Variational inference experiments To try and match the number of estimates of our Gibbs sampling experiments, we split each variational inference process into five independent runs and ran them for 100 passes each (or until convergence). Each run was initialized to split the data among five active components, though specifying the initial number of components did not seem to affect the number of active components at convergence. We varied the value of the hyper-parameter α for the DP over a grid of $\{0.5, 1.0, 2.0, 3.0\}$. Each pass through the dataset of 28876 points took approximately 1/6th of one wall-clock second, illustrating just how much faster variational inference is compared to collapsed Gibbs sampling. We modified the code provided as part of a Bayesian nonparametric library `bnpy` [54] to accommodate our code. We report the evidence lower bound of our models (ELBO) depicted in Figure 5.2.

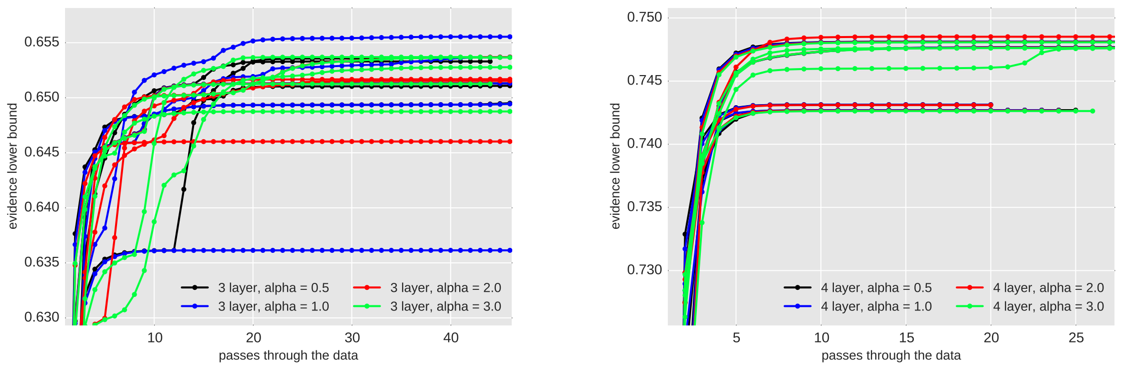


Figure 5.2: Trace plots of the ELBO $\mathcal{L}(q)$ on the marginal likelihood for select experiments of the DPMM fit by variational inference. We fit models using the data reduce by both the three layer SdA (left) and the four layer SdA (right). The models were each initialized to allocate data among 5 components, and differed only in their α hyper-parameter values.

The ELBO for models fit on the data reduced by the four layer SdA was consistently higher than that of models using data reduced by the three layer SdA. This result is encouraging as it supports the findings presented in Figure 4.10.

Amalgamated evidence for the active components We present our posterior estimates of the distribution over the number of active components for both methods. This was compiled by counting the reported number of components in the model across all Gibbs sampling chains and variational inference model epochs.

Importantly, there is ambiguity among the models over the number of components that best explains the data. Examining Figure 5.3, we see that even when aggregating the traces over many different models and inferring the number of components in different ways, there is still good support for between three and five components. This is a strong argument against using maximum likelihood to infer the number of components in one single model.

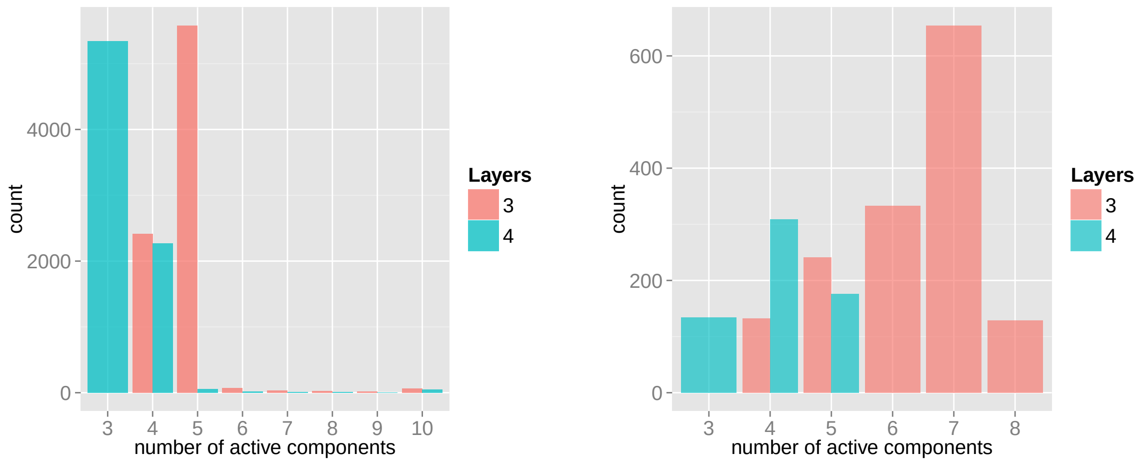


Figure 5.3: Histograms of the number of active components for both models fit by collapsed Gibbs sampling (left) and variational inference (right). We see that the Gibbs samplers strongly favour fewer components as compared to both variational inference models.

5.3 Profiling populations on RAD52 data set

Given the ambiguity over the number of components, we chose to fit several reference models. We chose the top performing methods by ELBO for each value of the DP hyper-parameter α, fit over both the 3 layer and 4 layer SdA reduced data sets. Models in hand, we then assigned each well a score based on the following pipeline:

Algorithm 3: Scoring Pipeline

```
input : Reduced whole screen data $D$, reference model
output: Distance-based score for each well
initialize well to gene map
extract and initialize reference model GMM
for well data $X_w$ in $D$ do
    compute posterior responsibilities under the model $p(\pi|W_x)$
    normalize responsibilities $r = \frac{1}{|X_w|} \times \sum_{rows} p(\pi|X_w)$
    compute symmetric KL score $s = \frac{KL(\pi_{ref}\|r)+KL(r\|\pi_{ref})}{2}$
end
```

As a control measure, we fit Gaussian mixture models using maximum likelihood for both reference datasets used to train the DPMMs. For each reference data set, we used the Bayesian information criterion to choose the optimal number of components and then fit full covariance Gaussian mixture models on the data. We scored each well in each control dataset in the same manner as for our other mixtures. Figure 5.4 shows the distribution of scores for the entire screen as a histogram for each DPMM trained on the three layer SdA reference set.

Figure 5.4 shows that the models have broad agreement in distribution of their scores. The control assigns lower scores to an additional small fraction of the screen. The four layer models share this broad similarity.

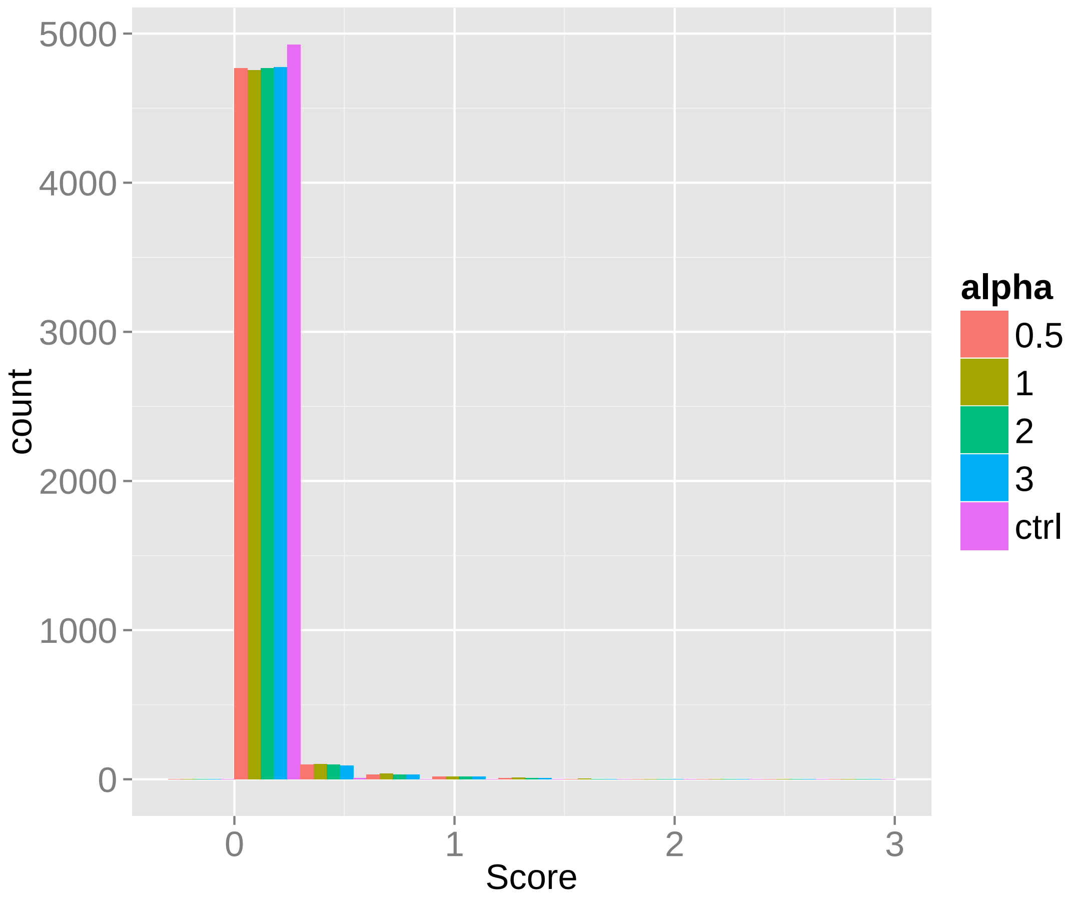


Figure 5.4: Scores for symmetric KL test of DPMMs and GMM control.

5.4 Results

Having used these DPMMs as a measure of closeness to a reference population, we investigated what sorts of genes scored highly and contrasted the results of the different models with each other. Our first measure was to see if the genes judged as outliers by the symmetric KL scores had any overlap with genes designated as hits under the supervised learning pipeline of Chapter 3. We took the top 10, 20 and 50 ORFs by symmetric KL score for each DPMM and control, and counted the number of ORFs that were also classified as hits in the single deletion condition.

The results in Table 5.1 shows little overlap between the two lists. This is not unexpected, since unsupervised methods model structure in the data. Without any incentive in the formulation of the loss function to favour variation related to foci, our method could not prioritize the scores for genes which displayed this type of variation. The similarity in the histogram distributions of Figure 5.4 motivated us to investigate the overlap of high scoring genes. Given the high degree of model similarity shown in the coarse-grained histogram view of Figure 5.4, we expected a greater degree of overlap among both families of models.

Encouragingly, a closer examination of the well images associated with the genes in the overlap yielded examples of cells possessing abnormal morphologies. We profile select images in Figure 5.6

Model	Top 10	Top 20	Top 50	SdA data
$\alpha = 0.5$	1	4	8	3 layer
$\alpha = 1.0$	0	3	9	3 layer
$\alpha = 2.0$	1	4	8	3 layer
$\alpha = 3.0$	1	4	9	3 layer
control	0	3	5	3 layer
$\alpha = 0.5$	1	3	5	4 layer
$\alpha = 1.0$	1	3	5	4 layer
$\alpha = 2.0$	1	3	5	4 layer
$\alpha = 3.0$	1	3	5	4 layer
control	1	4	10	4 layer

Table 5.1: Overlap between the ORF lists of outlier scoring models ranked descending by symmetric KL distance from a reference population, and the single mutant ranked score list according to combined Fisher's method score from Chapter three. The entries represent the count of how many ORFs that appear in the highest scoring 10, 20 or 50 for each outlier scoring model also appear as a hit in the single mutant foci screen.

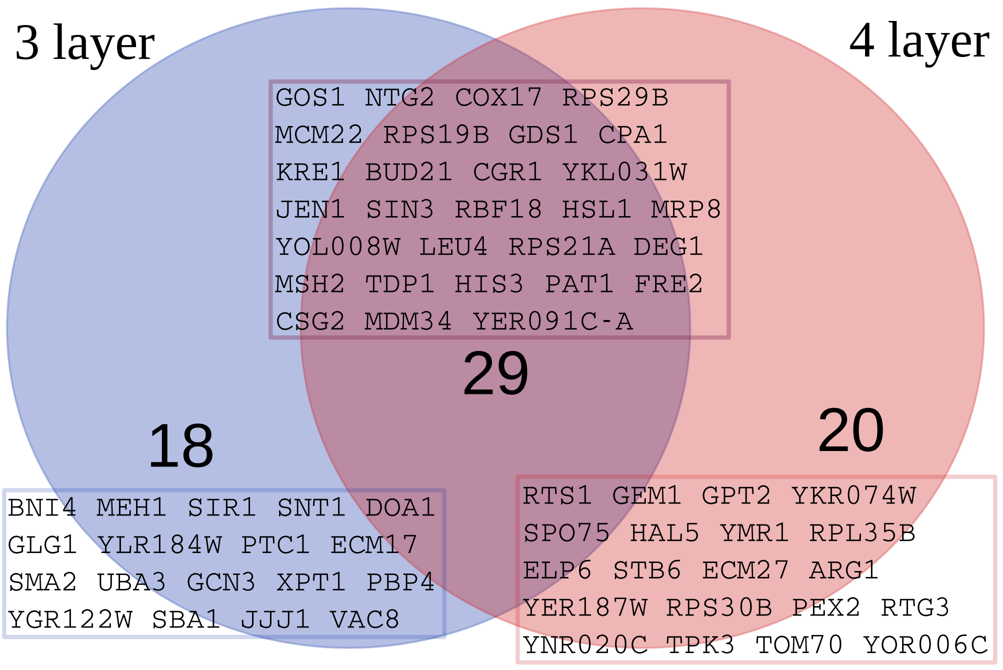


Figure 5.5: Top 50 ORFs of three versus four layer models. Overlapping ORFs appear at the intersection.

and discuss the phenotypes observed within. In the top left panel is a sample of a cell drawn from the BUD21 mutant which displays abnormal bud-neck formation, from which the gene name is derived [87]. In the bottom left panel is a sample of cells drawn from the CSG2 mutant population, which is an endoplasmic reticulum membrane protein that has been previously implicated as active when yeast is under replication stress [120]. In the lower right panel is a sample of cells from the MSH2 mutant population. The top-most cell in the picture displays a double-foci phenotype. This is of particular interest since it does not agree with the notion that multiple DSBs are recruited to the same DNA repair complex [70]. In the top right panel we display sample cells from the wild-type population as a control.

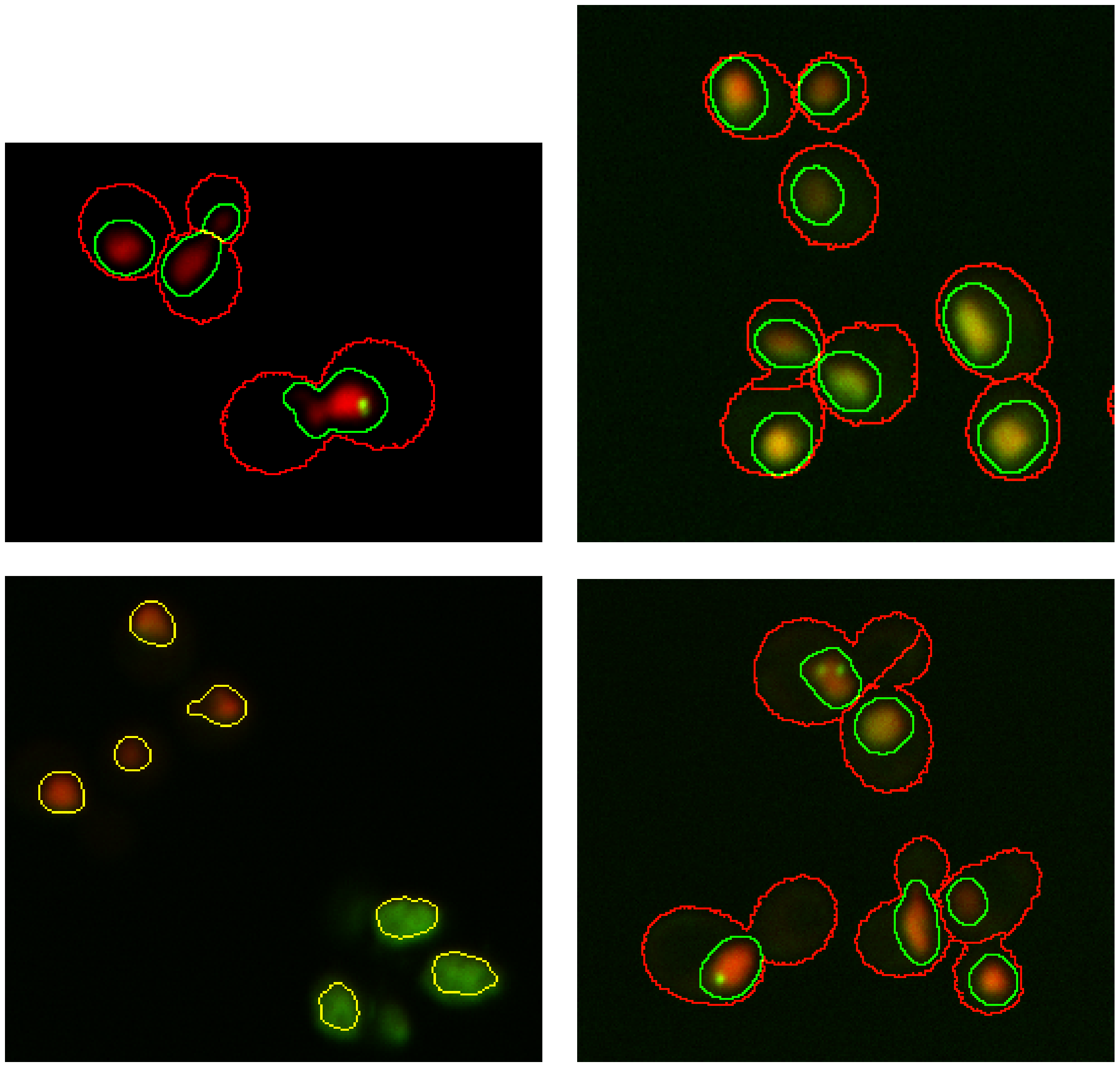


Figure 5.6: Sample cropped images from different genes designated as hits in the intersection of high ranked genes from both models. **Top left:** A sample from the mutant BUD21. **Top right:** A sample from wild-type cells acting as control. **Bottom left:** A sample from the mutant CSG2. **Bottom right:** A sample from the mutant MSH2.

In addition to sample successes, we highlight some instructive cases where our method assigned a high score to populations that display some irrelevant or noisy variation. Figure 5.7 displays two examples. On the left is a cropped sample from the mutant population SIN3. On the right is a cropped sample from the mutant population COX17. In both cases,m we see that segmentation has failed to the cell borders correctly. This would induce a large difference in the features measuring object size and shape, which could lead to both mutants being assigned high scores. Another obvious error is the presence of contaminants. In SIN3 we see what may be a bacterium (over-segmented object in green), while in COX17 we see a long diagonal streak of low intensity green that may be dust.

From our manual verification in Figures 5.6 and 5.7, we see that when the model assigns high scores to populations, they often display large differences from wild-type cells. We would also like to know what

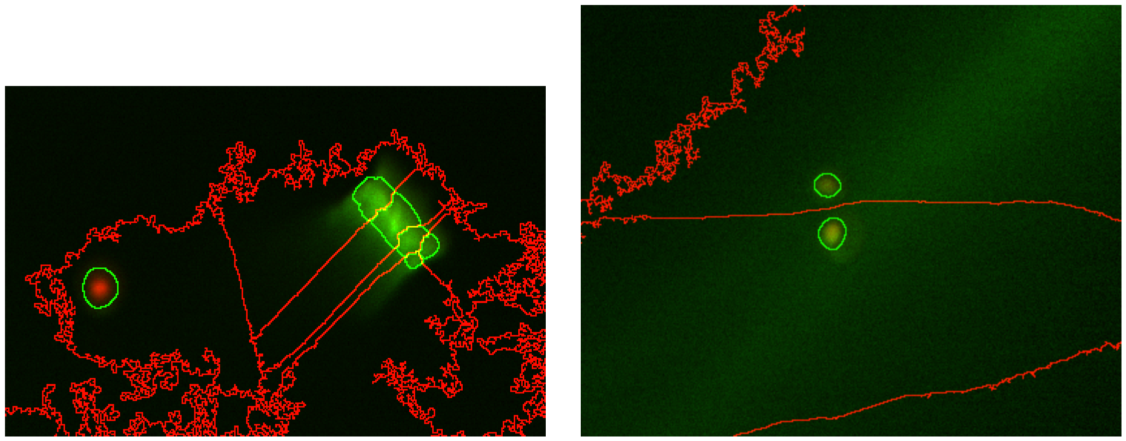


Figure 5.7: Sample cropped images from genes SIN3 (left) and COX17 (right) which represent false positives in our outlier detection search. Both images have clear flaws in the segmentation (red lines). Both also display the presence of either contamination (left) or foreign objects (right).

is common about the functions of genes that are assigned high scores. For a broader investigation of those genes in the intersection, we investigated them using a similar approach to the hit lists of Chapter 3, by visualizing them as a GeneMANIA network. We input this list of genes into the GeneMANIA website and allowed it to determine the weights and sources of evidence to form the network.

Here we highlight several sub-networks. One group, consisting of SIN3, MCM32, PAT1, and MDM34 are broadly involved in chromatin maintenance and remodeling. Another interesting group is MSH2, NTG2 and TDP1, which are genes implicated in DNA repair. Connected via genetic interaction, the high scores assigned to these genes suggests that the DPMM detects some of the signal focusing on DNA damage. A third group whose presence is easily explained is the BUD21 and HSL1 duo. The Saccharomyces Genome Database reports that deletion of these two genes results in abnormal budding patterns [23], which suggests that the distribution of cell morphology captured in the input features was also detected by the model. Finally, there were two interesting proteins of unknown function, MRP8 and GDS1. The protein expression levels of MRP8 are known to be up-regulated in response to DNA replication stress, and it is thought to undertake some function in transcription. Even less is known about GDS1, except that it is essential for growth in conditions where glycerol is the carbon source.

5.5 Discussion

This chapter introduced a model for detecting outliers at the population level in high content screening data, building on the dimensionality reduction method presented previously. We employed two different methods for training a DPMM and showed how the methods' complementary strengths helped us gain a more complete understanding of the model. Diagnostic results, in the form of a histogram over the number of components, showed that there was good reason to not rely on one standard mixture of Gaussians model fit by maximum likelihood, a disadvantage not shared by the more flexible nonparametric DPMM. For the population in each well of our screen, we evaluated the posterior responsibilities of the data given our reference models and used this to measure the symmetric difference in distribution from our reference models to characterize the degree to which each population differed from the reference. We were surprised that the difference in the score distribution between our model and the control Gaussian

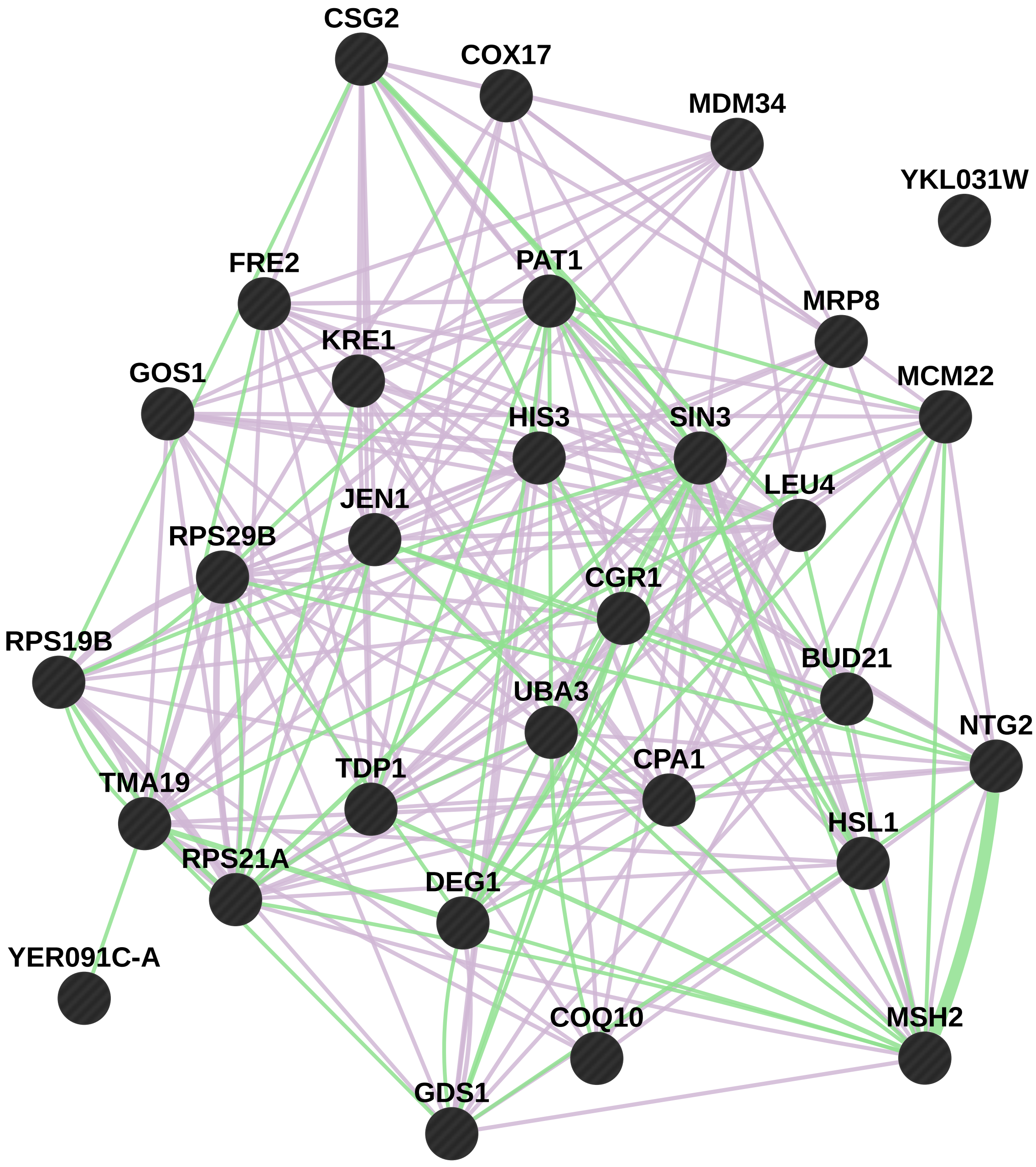


Figure 5.8: Intersection of the top 50 genes for top performing ELBO models from the four layer and three layer datasets. Magenta lines represent co-expression while green lines indicate genetic interaction.

mixture model was not dramatic. Manual investigation of select hits in Figure 5.6 provided evidence that high scoring genes do display abnormal morphology. However, Figure 5.7 showed that some of these genes were clearly false positives, as their high score were not biologically interesting and merely were indicative of sample contamination or a segmentation failure. Taken together, results from both figures suggest there are many ways in which the models' performance might be improved, beginning with the scoring method. The distribution of the population scores is influenced by the choice of KL divergence to measure differences in distribution. This ignores any other information we have about the population itself. Ignoring functional information or population size might decrease our power to detect anything beyond broad differences. As a simple improvement, population weights can be incorporated into the calculation of a Bregman divergence which represents the extent of difference we expect to observe from the reference.

Chapter 6

Conclusion

This thesis presented three methods that address the different problems that arise in the course of determining how observed differences in the distribution of cell phenotypes might be explained by genetic perturbations. Each of our methods was motivated by a different core problem in HCS analysis: automating the scoring of whole populations by classifying their cells, coping with the vast amount of high-dimensional data, and scoring populations in the absence of any labeled data. Human expert time is scarce, so the value of the time invested must be maximized.

6.1 Chapter three

In Chapter three, we presented a pipeline for processing HCS images that reduces the amount of manual work. Using cell object data represented as a bag of features along with a small subset of labels, our method produced a ranked list of genes for each experimental condition. Our gene list was strongly enriched for functions which revolve around the repair of DNA damage. The combined results from replicate screens empowered our method to uncover plausible new links for genes involved in double-strand break repair. In addition to the amount of data, a key contribution was qualifying how population size affected the reliability of estimated number of foci in the population. This affected how we normalized the raw scores and combined different replicates to ultimately rank each gene.

6.1.1 Revisiting performance across conditions and potential limitations

At the end of Chapter three, we contrasted the SVM classifier performance on sampled data from replicate one, where the training set was sourced, against sampled data from replicate three (cf. Figure 3.14). The observed difference in ROC led us to question the potential explanations of these disparities: was it was due to overfitting the training data or to a fundamental shift in the marginal distribution over covariates in the replicates? To try to explain the observed difference, we examined different aspects of the performance of classifiers across replicates in a series of experiments aimed at revealing the extent of covariate shift that exists between replicates of the single deletion condition.

First, we tried to quantify the degree of covariate shift among replicates of the single deletion collection by testing the degree to which data points from one replicate could be distinguished from another. For each pair of replicates, we randomly sampled 1000 cells and labeled them as being drawn from replicate one, two, or three (as appropriate). We then split each labeled replicate data into training and test sets

Run	One vs Two	One vs Three	Two vs Three
1	0.54	0.59	0.64
2	0.77	0.70	0.61
3	0.52	0.70	0.70
4	0.73	0.58	0.80
5	0.47	0.61	0.69
Average	**0.606**	**0.636**	**0.688**

Table 6.1: Area under the ROC curve for data provenance prediction of each replicate of the single deletion collection against each other replicate. The average AUC is reported in the last row.

and combined all of the training and test data separately. Using this unified training and test data, we trained a logistic regression classifier to predict the provenance of data point in the test set.

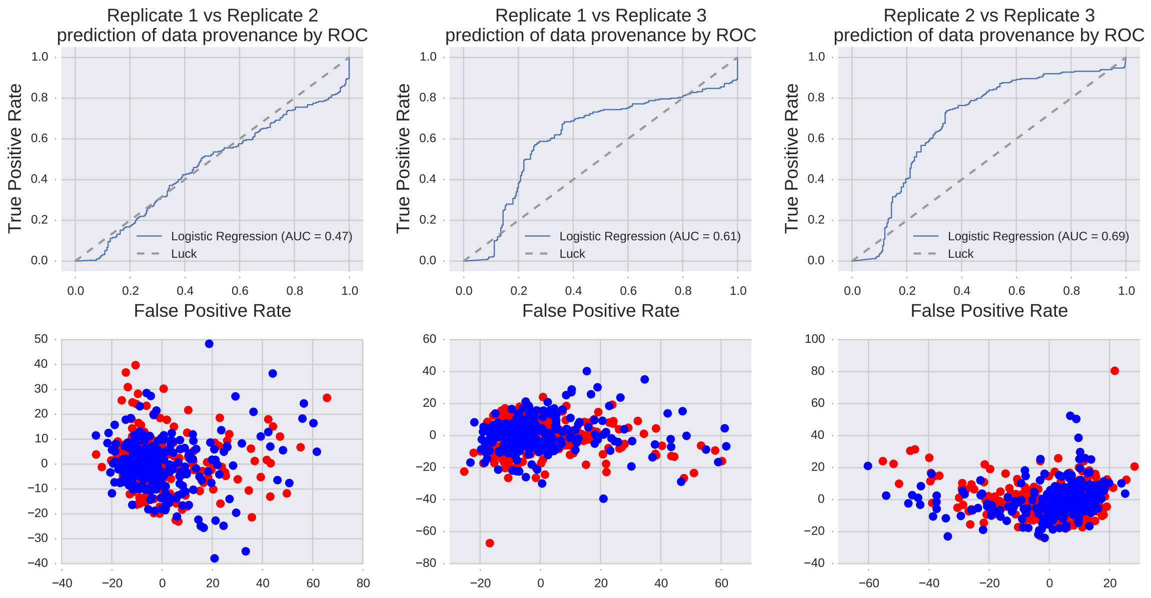


Figure 6.1: Top: ROC of a logistic regression classifier when predicting data provenance of one single deletion collection replicate versus another. Bottom: PCA plots in two dimensions of the sampled data from each replicate, coloured arbitrarily.

We repeated the experiment in Figure 6.1 five times and report the averaged AUC scores in Table 6.1 below:

If each replicate were drawn from the same distribution with unbiased sampling, then the classifier should produce an average ROC of 0.5. The degree to which each average ROC exceeds 0.5 can be interpreted as a measure of the degree of covariate shift between each replicate. These experiments establish the presence of covariate shift between each of the replicates; however, they do not tell us about how the errors induced via covariate shift are distributed. To delve into this, we examined the performance of classifiers using precision and recall to identify the distribution of errors (false positive versus false negative), and consider the risk incurred by each. In Figure 6.2, we plot the precision versus recall of each classifier, trained with data from replicate one, on test data drawn from each of the three replicates.

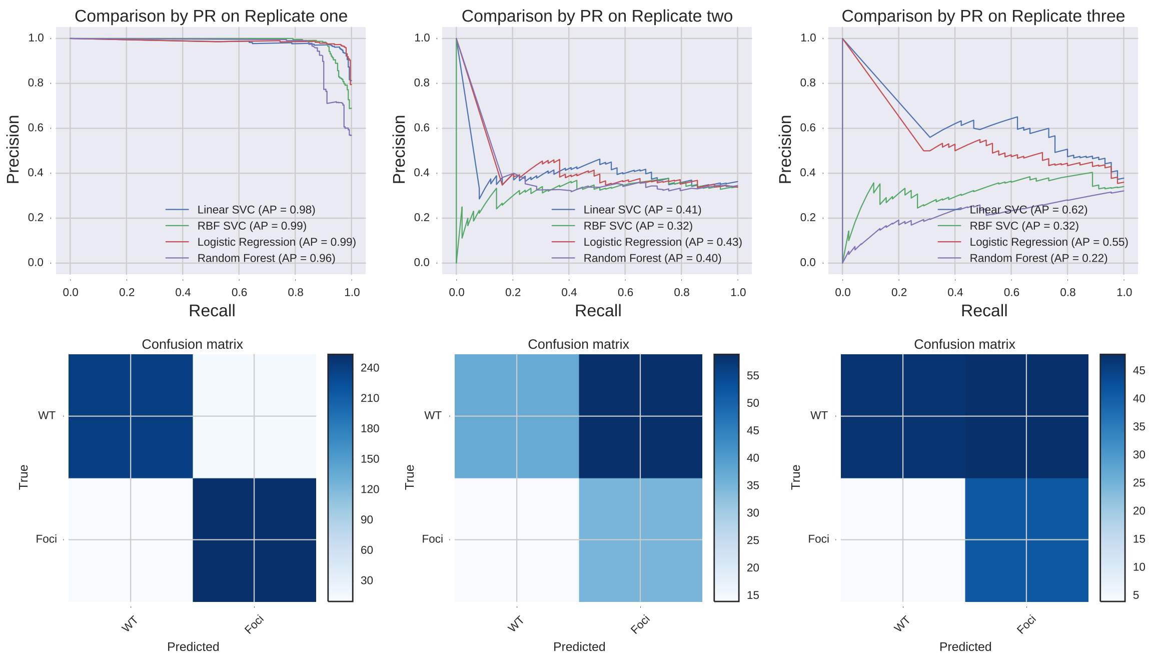


Figure 6.2: Precision versus recall of classifiers, trained on data from replicate one, on test data from each biological replicate in the single deletion condition. Top panels are precision-recall curves comparing the discriminative classifiers labeled in the legends. Bottom panels are the confusion matrices for the predicted labels of the linear kernel SVM on a test set from different replicates.

Below each curve, we plot the confusion matrix for the test. The performance declines on replicates two and three, as expected from Figure 3.14, but the distribution of errors is informative; the SVM is prone to making false positive errors rather than false negative errors. Finally, to see if this phenomenon was more widely present over all the different conditions, we randomly sampled approximately 500 cells from each condition, manually checked the labeling of each cell, and tabulated the results of the false positive versus false negative rates over each condition (Figure 6.3).

The differences in rates over the different conditions depicted in Figure 6.3 vary widely for false positives but almost not at all for false negatives. This means that we will tend to over-estimate the true foci rates of genes. Yet when considered in light of the pooled results over all assays of a given condition, which we demonstrated in Section 3.5.2, the method produces biologically sensible results. How can we explain this apparent contradiction?

One answer may be that our method for ranking scores combined across replicates tends to smooth out errors that would otherwise increase the rank of genes that have very high false positive rates. Another explanation may be that false positives are distributed in a way that does not alter rankings. Since we care about the rank of genes rather than the absolute scores, over-estimation will not have a large effect as long as the distribution of false positives is close to uniform. Using the single mutant population as a benchmark, the largest differences in false positive rates was observed in the single deletion along with sgs1 deletion screen. This is not unexpected; removing an active member of the homologous recombination pathway for repairing DNA damage should subject the cells to serious stress.

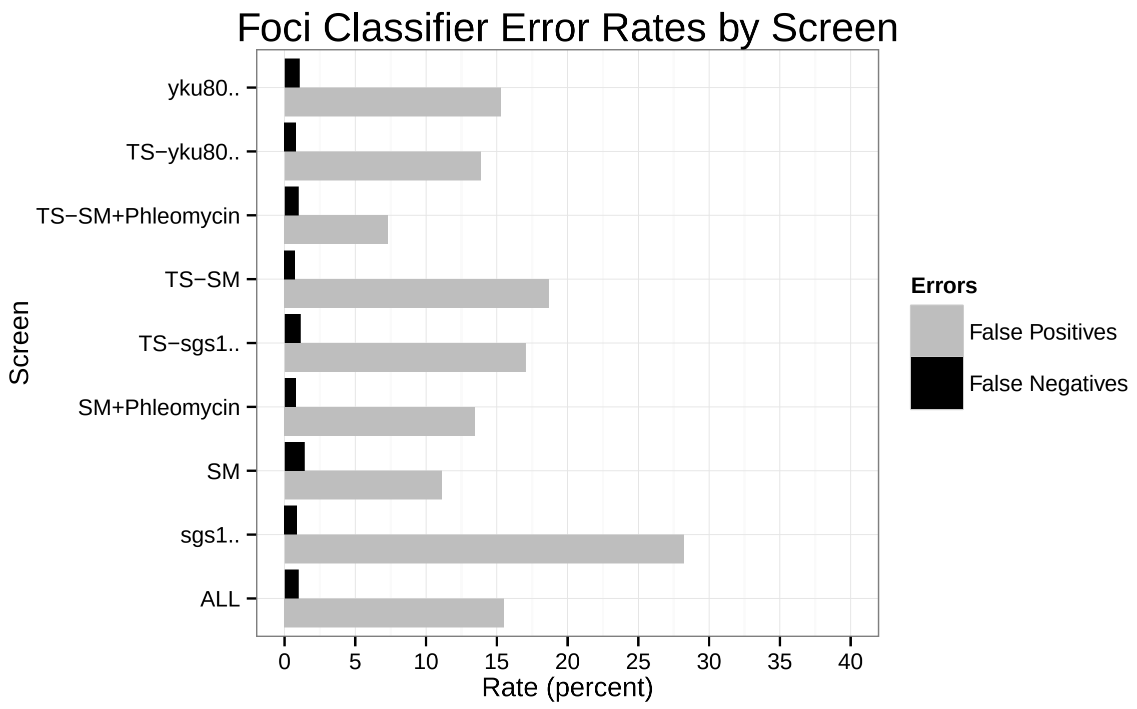


Figure 6.3: Type I and II errors for the foci classifer on randomly sampled cells for each of the eight different conditions, as well as the averaged rates for all conditions.

6.1.2 Possible extensions

A potential extension to this work is to have a scoring scheme to emphasize moderately high scoring genes. The very top hits were consistently genes whose function was directly related to DNA damage. Observing them as the highest ranked genes was evidence that our method worked, however it will not help us discover new biology. Many experiments have confirmed that genes such as $MMS1$ play a central role in DNA damage, so observing it as the number one gene in our list tells us little we did not already know. Much more interesting were genes that achieved a high score but whose annotations in SGD or GO did not suggest any explanation for the high rank. With this in mind, we might consider separating the scoring and ranking of genes into two different tasks in multi-task classification. One task would be to produce a ranking that is consistent across replicates and the other would be to gauge the surprise of that ranking in terms of the likelihood that this represents a new discovery.

6.2 Chapter four

In Chapter four, we presented and explored the use of deep autoencoders for dimensionality reduction of HCS data. Revisiting the technique of unsupervised pre-training allowed us to build a complex model layer by layer without using any labeled data, and subsequently minimizing the reconstruction error via mini-batch stochastic gradient descent allowed us to achieve good performance relative to other comparable methods. Stacking de-noising autoencoders has been explored in many different contexts, but almost always as a means to initialize a different model (usually a multi-layer perceptron) for some classification task. To the best of our knowledge, this is the first instance where a stacked de-noising autoencoder has been trained purely unsupervised and applied to dimensionality reduction. We explored the effect on performance as a result of the many hyper-parameters included in the model. However, we found that the greatest gain in performance was due mostly to properly tuning the learning rate and rescaling the gradient update for each dimension with the adaptive gradient descent [133]. The computationally intensive training, dominated primarily by dense matrix multiplication, was accelerated using GPUs. Also invaluable was generation of gradient updates by symbolic differentiation provided by Theano [13, 9]. Performing gradient descent on a function of millions of parameters opens up the possibility of millions of bugs; it is imperative to know that it is being calculated correctly.

6.2.1 Limitations and considerations

It has long been established that deep neural network models are difficult to train. While pre-training and more aggressive regularization schemes have made it possible to achieve better results by ensuring the proper flow of gradient information, many limitations remain in the models we have used here. First, the use of a sigmoid non-linearity forces the activation of each unit into $[0, 1]$. While this simplifies training, since it bounds the error that can be achieved during reconstruction of intermediate representations, it imposes all the burden of decoding the penultimate representation from $[0, 1]$ back into $\Re$, which requires the W parameter matrix to be extremely sensitive to small variations in the input.
Second, the method requires a very large amount of data for training. Without hundreds of thousands of examples on which to train, this method will almost surely lead to over-fitting due to the large number of parameters present. Though over-fitting may be mitigated by new regularization techniques, such as dropout, the extent to which dropout can counteract the potential for over-fitting a large number of parameters with a small dataset is not well characterized.
Finally, it is not known how these models are affected by covariate shift, such as shown in Figure 6.1. Covariate shift can be viewed as a particular form of importance sampling [111]. If the parametric forms of the training and test distributions were known, sampling weights for each element of each replicate could be calculated exactly, and we could sample mini-batches for training SdA models that would minimize covariate shift. This is usually not the case, so weights have to be estimated directly from the replicates [74, 14, 52].

6.2.2 Possible extensions

Perhaps the most immediate extension of an SdA model for dimensionality reduction would be a smarter and more principled method for tuning the many hyper-parameters involved in training. The number of units per layer, the amount of noise injected at each layer during pre-training, and the schedule for decaying momentum when fine-tuning are all hyper-parameters that we tuned in isolation, each via a time-consuming grid-search. Utilizing the recent developments in modeling error surfaces for automatic hyper-parameter tuning using Bayesian optimizaiton, such as the Spearmint algorithm of Snoek et al. [**?**], would greatly simplify training, would improve performance, and would make these models more extensible to problems in similar domains.

6.3 Chapter five

The SdA models presented in Chapter five were used to produce a low-dimensional encoding of our data, which we used for the task of outlier detection in Chapter five. We showed that using a more flexible mixture model gave us greater assurance that we were not risking a mistake by choosing an inappropriate capacity for our model. We explored two different ways of inferring the model parameters, opting in the end for variational inference based on the added bonus that the training objective was also a sound basis for model comparison. We assigned a score to every population in the dataset, which measured the difference in distribution under our DPMM against a randomly sampled reference population using the KL divergence. Though the overall distribution of scores was very similar compared to that of a control model, the act of developing and testing the model revealed opportunities for improvement.

6.3.1 Limitations and considerations

While there was reasonable overlap in the scores for top 50 genes among the two models compared in Figure 5.8, our manual inspection in Figures 5.6 and 5.7 show that the model scores can be dominated by large non-biological variation. We suspect that the probability of detection might be further improved by using an ensemble of models. Ensembles might be usefully applied here given that they have been shown to boost performance in many other tasks and to increase robustness in the presence of noise. Another advancement would be to treat the posterior responsibilities evaluated under the reference model as a signature of that population. As mentioned in the discussion of Chapter five, the KL divergence is indifferent to how differences in distribution arise. By contrast, a more fine-grained way to measure commonality among these signatures, such as hierarchical clustering, might allow us to detect more informative patterns of deviation common to different populations. Lastly, we feel that the degree of overlapping parameters of the models in Chapter five shows that while phenotypes may be discernible by eye, they are difficult to detect from a bag of features. Cells displaying different phenotypes will still share a large overlap for many parameters: they will have little variation in size and shape, for example. With this observation in mind, we question whether a model that induces a partition over the data is the most appropriate way to characterize its structure. Other nonparametric Bayesian models, such as the Indian Buffet Process among others, would still offer a nonparametric characterization of the data while allowing for data points to overlap membership in the latent factors representing phenotypes.

6.3.2 Possible extensions

An appealing alternative to constructing a single non-parametric reference model would be to attempt to cluster all replicates of each population simultaneously using a hierarchical version of a Dirichlet-process Gaussian-mixture model [117]. Instead of considering every gene separately, a hierarchical Dirichlet process treats the cells in each population and replicate as specific noisy realizations of a latent mixture of cellular phenotypes. Rather than merely allowing us to expand to a more complete reference set, this model would allow the reference mixture distribution to be learned as part of the model. This would not only eliminate tricky sampling issues raised by the inclusion of an observed reference set, but it would also allow all replicates to be clustered simultaneously.

6.4 Concluding Remarks

From a broader perspective, our motivating question in this thesis has been how to build better computational tools for explaining observed variation in cell phenotype data as a consequence of genetic effects. Taken together, the success of our experiments in Chapter three and modest results in Chapter five suggest the best way to accomplish this involves leveraging large amounts of unlabeled data while maximizing use of available labeled data. Since the cost of generating labeled data in this domain is likely to remain high, we believe that putting more effort into developing models that transfer previous knowledge contained in data from similar studies is the most likely approach to lead to significant further advances in HCS. Research into these methods, known as transfer learning or domain generalization in machine learning, offers promising leads for fields like HCS, where labels are likely to remain scarce.

Bibliography

[1] Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Muller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gäel Varoquaux. Machine Learning for Neuroimaging with Scikit-Learn. *Frontiers in Neuroinformatics*, 8, December 2014.

[2] Ram Sunanda J Abramoff M.D., Magalhães Paulo J. Image processing with ImageJ. *Biophotonics international*, 11(7):36–42, 2004.

[3] David Alvaro, Michael Lisby, and Rodney Rothstein. Genome-wide analysis of Rad52 foci reveals diverse mechanisms impacting recombination. *PLoS genetics*, 3(12):e228, December 2007.

[4] Charles E Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The annals of statistics*, pages 1152–1174, 1974.

[5] Mustapha Aouida and Dindial Ramotar. Bleomycin transport holds the key for improved anti-cancer therapy. *Cancer Therapy*, 4:171–182, 2006.

[6] Chris Bakal, John Aach, George Church, and Norbert Perrimon. Quantitative morphological signatures define local signaling networks regulating cell morphology. *Science (New York, N.Y.)*, 316(5832):1753–6, June 2007.

[7] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, January 1989.

[8] Jacqueline H Barlow and Rodney Rothstein. Timing is everything: cell cycle control of Rad52. *Cell division*, 5(1):7, January 2010.

[9] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Technical report, Universite de Montreal, Lake Tahoe, 2012.

[10] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: a review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–828, August 2013.

[11] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In B Schölkopf, J Platt, and T Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, Cambridge, MA, 2007.

[12] Yoshua Bengio, Jean-Francois Paiement, Pascal Vincent, Olivier Delalllaux, Nicholas Le Roux, and Marie Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and Spectral Clustering. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in neural information processing systems 16: proceedings of the 2003 conference*, page 1621. MIT Press, 2004.

[13] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a {CPU} and {GPU} Math Expression Compiler. In *Proceedings of the Python for Scientific Computing Conference ({SciPy})*, June 2010.

[14] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative Learning Under Covariate Shift. *Journal of Machine Learning Research*, 10:2137—-2155, September 2009.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, August 2006.

[16] David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, March 2006.

[17] M V Boland, M K Markey, and R F Murphy. Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images. *Cytometry*, 33(3):366–75, November 1998.

[18] Michael Boutros, Lígia P Brás, and Wolfgang Huber. Analysis of cell-based RNAi screens. *Genome biology*, 7(7):R66, January 2006.

[19] Christine Brideau, Bert Gunter, Bill Pikounis, and Andy Liaw. Improved statistical methods for hit selection in high-throughput screening. *Journal of biomolecular screening*, 8(6):634–47, December 2003.

[20] Anne E Carpenter, Thouis R Jones, Michael R Lamprecht, Colin Clarke, In Han Kang, Ola Friman, David A Guertin, Joo Han Chang, Robert A Lindquist, Jason Moffat, Polina Golland, and David M Sabatini. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology*, 7(10):R100, January 2006.

[21] Chih-Chung Chang and Chih-Jen Lin. LIBSVM. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, April 2011.

[22] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, September 2006.

[23] J Michael Cherry, Eurie L Hong, Craig Amundsen, Rama Balakrishnan, Gail Binkley, Esther T Chan, Karen R Christie, Maria C Costanzo, Selina S Dwight, Stacia R Engel, Dianna G Fisk, Jodi E Hirschman, Benjamin C Hitz, Kalpana Karra, Cynthia J Krieger, Stuart R Miyasato, Rob S Nash, Julie Park, Marek S Skrzypek, Matt Simison, Shuai Weng, and Edith D Wong. Saccharomyces Genome Database: the genomics resource of budding yeast. *Nucleic acids research*, 40(Database issue):D700–5, January 2012.

[24] Luis Pedro Coelho, Estelle Glory-Afshar, Joshua Kangas, Shannon Quinn, Aabid Shariff, and Robert F Murphy. Principles of Bioimage Informatics : Focus on Machine Learning of Cell Patterns. *IEEE Engineering in Medicine and Biology Magazine*, pages 8–18, 2010.

[25] Luis Pedro Coelho, Tao Peng, and Robert F Murphy. Quantifying the distribution of probes between subcellular locations using unsupervised pattern unmixing. *Bioinformatics*, 26(12):i7–12, 2010.

[26] Luis Pedro Coelho, Tao Peng, and Robert F Murphy. Quantifying the distribution of probes between subcellular locations using unsupervised pattern unmixing. *Bioinformatics (Oxford, England)*, 26(12):i7–12, June 2010.

[27] A A Cohen, N Geva-Zatorsky, E Eden, M Frenkel-Morgenstern, I Issaeva, A Sigal, R Milo, C Cohen-Saidon, Y Liron, Z Kam, L Cohen, T Danon, N Perzov, and U Alon. Dynamic proteomics of individual cancer cells in response to a drug. *Science (New York, N.Y.)*, 322(5907):1511–6, December 2008.

[28] Michael Costanzo, Anastasia Baryshnikova, Jeremy Bellay, Yungil Kim, Eric D. Spear, Carolyn S. Sevier, Huiming Ding, Judice L.Y. Koh, Kiana Toufighi, Sara Mostafavi, Jeany Prinz, Robert P. St. Onge, Benjamin VanderSluis, Taras Makhnevych, Franco J. Vizeacoumar, Solmaz Alizadeh, Sondra Bahr, Renee L. Brost, Yiqun Chen, Murat Cokol, Raamesh Deshpande, Zhijian Li, Zhen-Yuan Lin, Wendy Liang, Michaela Marback, Jadine Paw, Bryan-Joseph San Luis, Ermira Shuteriqi, Amy Hin Yan Tong, Nydia van Dyk, Iain M. Wallace, Joseph A. Whitney, Matthew T. Weirauch, Guoqing Zhong, Hongwei Zhu, Walid A. Houry, Michael Brudno, Sasan Ragibizadeh, Balázs Papp, Csaba Pál, Frederick P. Roth, Guri Giaever, Corey Nislow, Olga G. Troyanskaya, Howard Bussey, Gary D. Bader, Anne-Claude Gingras, Quaid D. Morris, Philip M. Kim, Chris A. Kaiser, Chad L. Myers, Brenda J. Andrews, and Charles Boone. The Genetic Landscape of a Cell. *Science*, 327(5964):425 –431, January 2010.

[29] Andrew Cron, Cécile Gouttefangeas, Jacob Frelinger, Lin Lin, Satwinder K Singh, Cedrik M Britten, Marij J P Welters, Sjoerd H van der Burg, Mike West, and Cliburn Chan. Hierarchical modeling for rare event detection and cell subset alignment across flow cytometry samples. *PLoS computational biology*, 9(7):e1003130, January 2013.

[30] Gaudenz Danuser. Computer Vision in Cell Biology. *Cell*, 147(5):973–978, November 2011.

[31] V. De Silva, J.B. Joshua B Tenenbaum, and Vin De Silva. Global versus local methods in nonlinear dimensionality reduction. *Advances in Neural Information Processing Systems 15*, 15(Figure 2):705–712, 2003.

[32] A.P. Dempster, N. M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[33] Laurens der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[34] Jonas F Dorn, Gaudenz Danuser, and Ge Yang. Computational processing and analysis of dynamic fluorescence image data. *Methods in cell biology*, 85:497–538, January 2008.

[35] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Technical report, University of California at Berkeley, Berkeley, February 2010.

[36] Murat Dundar, Ferit Akova, Halid Z Yerebakan, and Bartek Rajwa. A non-parametric Bayesian model for joint cell clustering and cluster matching: identification of anomalous sample phenotypes with random effects. *BMC bioinformatics*, 15(1):314, January 2014.

[37] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014.

[38] Florian Fuchs, Gregoire Pau, Dominique Kranz, Oleg Sklyar, Christoph Budjan, Sandra Steinbrink, Thomas Horn, Angelika Pedal, Wolfgang Huber, and Michael Boutros. Clustering phenotype populations by genome-wide RNAi and multiparametric imaging. *Mol Syst Biol*, 6(370), June 2010.

[39] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, November 1984.

[40] Samuel J. Gershman and David M. Blei. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, February 2012.

[41] Kenneth A Giuliano, Patricia A Johnston, Albert Gough, and D Lansing Taylor. Systems cell biology based on high-content screening. *Methods in enzymology*, 414:601–19, January 2006.

[42] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. *Journal of Machine Learning Research*, 15:315–323, 2011.

[43] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, March 2003.

[44] Jihun Ham, Daniel D. Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Twenty-first international conference on Machine learning - ICML '04*, page 47, New York, New York, USA, July 2004. ACM Press.

[45] Nicholas A Hamilton and Rohan D Teasdale. Visualizing and clustering high throughput subcellular localization imaging. *BMC Bioinformatics*, 9(1):81, 2008.

[46] Louis-François Handfield, Yolanda T. Chong, Jibril Simmons, Brenda J. Andrews, and Alan M. Moses. Unsupervised Clustering of Subcellular Protein Expression Patterns in High-Throughput Microscopy Images Reveals Protein Complexes and Functional Relationships between Proteins. *PLoS Computational Biology*, 9(6):e1003085, June 2013.

[47] L H Hartwell, J J Hopfield, S Leibler, and A W Murray. From molecular to modular cell biology. *Nature*, 402(6761 Suppl):C47–52, December 1999.

[48] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[49] Maureen E Hillenmeyer, Eula Fung, Jan Wildenhain, Sarah E Pierce, Shawn Hoon, William Lee, Michael Proctor, Robert P St Onge, Mike Tyers, Daphne Koller, Russ B Altman, Ronald W Davis, Corey Nislow, and Guri Giaever. The chemical genomic portrait of yeast: uncovering a phenotype for all genes. *Science (New York, N.Y.)*, 320(5874):362–5, April 2008.

[50] G E Hinton and R R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science (New York, N.Y.)*, 313(5786):504–7, July 2006.

[51] Geoffrey E Hinton, Simon Osindero, and Yee-Whye W Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[52] Jiayuan Huang, Alexander J Smola, Arthur Gretton, Karsten M Borgwardt, and Bernhard Schölkopf. Correcting Sample Selection Bias by Unlabeled Data. In B Schölkopf, J Platt, and T Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 601–608. MIT Press, Cambridge, MA, 2007.

[53] Michael C Hughes and Erik Sudderth. Memoized Online Variational Inference for Dirichlet Process Mixture Models. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1133–1141. Curran Associates, Inc., 2013.

[54] Michael C. Hughes and Erik B. Sudderth. bnpy: Reliable and scalable variational inference for Bayesian nonparametric models. In *3rd NIPS Workshop on Probabilistic Programming*, Montréal, 2014.

[55] Nathalie Japkowicz, Stephen José Hanson, and Mark A. Gluck. Nonlinear Autoassociation Is Not Equivalent to PCA. *Neural Computation*, 12(3):531–545, March 2000.

[56] Thouis R Jones, In Han Kang, Douglas B Wheeler, Robert A Lindquist, Adam Papallo, David M Sabatini, Polina Golland, and Anne E Carpenter. CellProfiler Analyst: data exploration and analysis software for complex image-based screens. *BMC bioinformatics*, 9(1):482, January 2008.

[57] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

[58] Lee Kamentsky, Thouis R Jones, Adam Fraser, Mark-Anthony Bray, David J Logan, Katherine L Madden, Vebjorn Ljosa, Curtis Rueden, Kevin W Eliceiri, and Anne E Carpenter. Improved structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. *Bioinformatics (Oxford, England)*, 27(8):1179–80, April 2011.

[59] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[60] Lawrence K. Saul Kilian Q. Weinberger. Unsupervised Learning of Image Manifolds by Semidefinite Programming, June 2004.

[61] Alex Krizhevsky and Geoffrey E. Hinton. *Learning Multiple Layers of Features from Tiny Images*. Msc thesis, University of Toronto, Toronto, 2009.

[62] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 473–480, New York, New York, USA, June 2007. ACM Press.

[63] Steffen Lawo, Monica Hasegan, Gagan D Gupta, and Laurence Pelletier. Subdiffraction imaging of centrosomes reveals higher-order organizational features of pericentriolar material. *Nature cell biology*, 14(11):1148–58, October 2012.

[64] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II—-97. IEEE, 2004.

[65] Seungtaek Lee and Bonnie J Howell. High-content screening: emerging hardware and software technologies. *Methods in enzymology*, 414:468–83, January 2006.

[66] Sasha F Levy and Mark L Siegal. Network hubs buffer environmental variation in Saccharomyces cerevisiae. *PLoS biology*, 6(11):e264, November 2008.

[67] Gang Lin, Umesh Adiga, Kathy Olson, John F Guzowski, Carol A Barnes, and Badrinath Roysam. A hybrid 3D watershed algorithm incorporating gradient cues and object models for automatic segmentation of nuclei in confocal image stacks. *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, 56(1):23–36, November 2003.

[68] JENNIFER LIPPINCOTT-SCHWARTZ, NIHAL ALTAN-BONNET, and GEORGE H. PATTERSON. Photobleaching and photoactivation: following protein dynamics in living cells. *Nature Cell Biology*, 5(Suppl), 2003.

[69] Michael Lisby, Jacqueline H Barlow, Rebecca C Burgess, and Rodney Rothstein. Choreography of the DNA damage response: spatiotemporal relationships among checkpoint and repair proteins. *Cell*, 118(6):699–713, September 2004.

[70] Michael Lisby, Uffe H Mortensen, and Rodney Rothstein. Colocalization of multiple DNA double-strand breaks at a single Rad52 repair centre. *Nature cell biology*, 5(6):572–7, June 2003.

[71] Michael Lisby and Rodney Rothstein. DNA damage checkpoint and repair centers. *Current opinion in cell biology*, 16(3):328–34, June 2004.

[72] Vebjorn Ljosa and A.E. Carpenter. Introduction to the quantitative analysis of two-dimensional fluorescence microscopy images for cell-based screening. *PLoS computational biology*, 5(12):e1000603, December 2009.

[73] L.H. Lit-Hsin Loo, Lani F L.F. Wu, and Steven J S.J. Altschuler. Image-based multivariate profiling of drug responses from single cells. *Nature methods*, 4(5):445–454, May 2007.

[74] Marco Loog. Nearest neighbor-based importance weighting. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, September 2012.

[75] Eugen Lounkine, Michael J Keiser, Steven Whitebread, Dmitri Mikhailov, Jacques Hamon, Jeremy L Jenkins, Paul Lavan, Eckhard Weber, Allison K Doak, Serge Côté, Brian K Shoichet, and Laszlo Urban. Large-scale prediction and testing of drug activity on side-effect targets. *Nature*, 486(7403):361–7, June 2012.

[76] Nathalie Malo, James A Hanley, Sonia Cerquozzi, Jerry Pelletier, and Robert Nadon. Statistical practice in high-throughput screening data analysis. *Nature biotechnology*, 24(2):167–75, February 2006.

[77] Alexandre Matov, Kathryn Applegate, Praveen Kumar, Claudio Thoma, Wilhelm Krek, Gaudenz Danuser, and Torsten Wittmann. Analysis of microtubule dynamic instability using a plus-end growth marker. *Nature Methods*, 7(9):761–768, August 2010.

[78] Sean G Megason and Scott E Fraser. Imaging in systems biology. *Cell*, 130(5):784–95, September 2007.

[79] P. Mitra, C. A. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity, March 2002.

[80] Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, January 2012.

[81] Robert F Murphy. Communicating subcellular distributions. *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, 77(7):686–92, July 2010.

[82] Iain Murray. *Advances in {M}arkov chain {M}onte {C}arlo methods.* {P}h{D} thesis, Gatsby computational neuroscience unit, University College London, 2007.

[83] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, Dordecht, 1998.

[84] Radford M. Neal. Probabilistic Inference using Markov Chain Monte Carlo Methods'. Technical report, University of Toronto, Toronto, 1993.

[85] Radford M. Neal. Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics*, 9(2), February 2000.

[86] Beate Neumann, Thomas Walter, Jean-Karim Hériché, Jutta Bulkescher, Holger Erfle, Christian Conrad, Phill Rogers, Ina Poser, Michael Held, Urban Liebel, Cihan Cetin, Frank Sieckmann, Gregoire Pau, Rolf Kabbe, Annelie Wünsche, Venkata Satagopam, Michael H. A. Schmitz, Catherine Chapuis, Daniel W. Gerlich, Reinhard Schneider, Roland Eils, Wolfgang Huber, Jan-Michael Peters, Anthony A. Hyman, Richard Durbin, Rainer Pepperkok, and Jan Ellenberg. Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature*, 464(7289):721–727, April 2010.

[87] L. Ni and M. Snyder. A Genomic Study of the Bipolar Bud Site Selection Pattern in Saccharomyces cerevisiae. *Molecular Biology of the Cell*, 12(7):2147–2170, July 2001.

[88] Yoshikazu Ohya, Jun Sese, Masashi Yukawa, Fumi Sano, Yoichiro Nakatani, Taro L Saito, Ayaka Saka, Tomoyuki Fukuda, Satoru Ishihara, Satomi Oka, Genjiro Suzuki, Machika Watanabe, Aiko Hirata, Miwaka Ohtani, Hiroshi Sawai, Nicolas Fraysse, Jean-Paul Latgé, Jean M François, Markus Aebi, Seiji Tanaka, Sachiko Muramatsu, Hiroyuki Araki, Kintake Sonoike, Satoru Nogami, and Shinichi Morishita. High-dimensional and large-scale phenotyping of yeast mutants. *Proceedings of the National Academy of Sciences of the United States of America*, 102(52):19015–20, December 2005.

[89] Hanchuan Peng. Bioimage informatics: a new area of engineering biology. *Bioinformatics (Oxford, England)*, 24(17):1827–36, September 2008.

[90] Hanchuan Peng, Phuong Chung, Fuhui Long, Lei Qu, Arnim Jenett, Andrew M Seeds, Eugene W Myers, and Julie H Simpson. BrainAligner: 3D registration atlases of Drosophila brains. *Nature methods*, 8(6):493–500, July 2011.

[91] Hanchuan Peng, Zongcai Ruan, Fuhui Long, Julie H Simpson, and Eugene W Myers. V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nature biotechnology*, 28(4):348–53, May 2010.

[92] Zachary E Perlman, Michael D Slack, Yan Feng, Timothy J Mitchison, Lani F Wu, and Steven J Altschuler. Multidimensional drug profiling by automated microscopy. *Science (New York, N.Y.)*, 306(5699):1194–8, November 2004.

[93] Iulian Pruteanu-Malinici, Daniel L Mace, and Uwe Ohler. Automatic Annotation of Spatial Expression Patterns via Sparse Bayesian Factor Models. *PLoS Computational Biology*, 7(7):16, 2011.

[94] Arjun Raj, Patrick van den Bogaard, Scott A Rifkin, Alexander van Oudenaarden, and Sanjay Tyagi. Imaging individual mRNA molecules using multiple singly labeled probes. *Nature methods*, 5(10):877–9, October 2008.

[95] M RANZATO, P TAYLOR, J HOUSE, R FLAGAN, Y LECUN, and P PERONA. Automatic recognition of biological particles in microscopic images. *Pattern Recognition Letters*, 28(1):31–39, January 2007.

[96] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient Learning of Sparse Representations with an Energy-Based Model. In B Schölkopf, J Platt, and T Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1137–1144. MIT Press, Cambridge, MA, 2007.

[97] Carl Edward Rasmussen. The Infinite Gaussian Mixture Model. In S.A. et al. Solla, editor, *Advances in information processing systems 12*, pages 554–560. MIT Press, 2000.

[98] Annemie Ribbens, Frederik Maes, Dirk Vandermeulen, and Paul Suetens. Semisupervised probabilistic clustering of brain MR images including prior clinical information. In *MCV'10 Proceedings of the 2010 international MICCAI conference on Medical computer vision: recognition techniques and applications in medical imaging*, pages 184–194, September 2010.

[99] Nitzan Rimon and Maya Schuldiner. Getting the whole picture: combining throughput with content in microscopy. *Journal of cell science*, 124(Pt 22):3743–51, November 2011.

[100] Karsten Rodenacker and Ewert Bengtsson. A feature set for cytometry on digitized microscopic images. *Analytical cellular pathology : the journal of the European Society for Analytical Cellular Pathology*, 25(1):1–36, January 2003.

[101] Gustavo K Rohde, Alexandre J S Ribeiro, Kris N Dahl, and Robert F Murphy. Deformation-based nuclear morphometry: capturing nuclear shape variation in HeLa cells. *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, 73(4):341–50, April 2008.

[102] Andrew Rosenberg and Julia Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[103] S. T. Roweis. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, December 2000.

[104] P.K Sahoo, S Soltani, and A.K.C Wong. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2):233–260, February 1988.

[105] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[106] Lawrence K Saul and Sam T Roweis. Think Globally , Fit Locally : Unsupervised Learning of Low Dimensional Manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[107] LK Saul, KQ Weinberger, JH Ham, and F Sha. Spectral methods for dimensionality reduction. In Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien, editors, *Semisupervised Learning*, chapter 16, pages 293–306. MIT Press, 2006.

[108] Lothar Schermelleh, Rainer Heintzmann, and Heinrich Leonhardt. A guide to super-resolution fluorescence microscopy. *The Journal of cell biology*, 190(2):165–75, July 2010.

[109] Lior Shamir, John D. Delaney, Nikita Orlov, D. Mark Eckley, and Ilya G. Goldberg. Pattern Recognition Software and Techniques for Biological Image Analysis. *PLoS Comput Biol*, 6(11):e1000974, November 2010.

[110] Aabid Shariff, Joshua Kangas, Luis Pedro Coelho, Shannon Quinn, and Robert F Murphy. Automated image analysis for high-content screening and analysis. *Journal of biomolecular screening*, 15(7):726–34, August 2010.

[111] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, October 2000.

[112] M. D. Slack, E. D. Martinez, L. F. Wu, and S. J. Altschuler. Characterizing heterogeneous cellular responses to perturbations. *Proceedings of the National Academy of Sciences*, 105(49):19306–19311, December 2008.

[113] C.L. Smith. Basic confocal microscopy. *Current Protocols in Molecular Biology*, 2008.

[114] R. Socher. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, June 2009.

[115] Jason R Swedlow, Ilya Goldberg, Erik Brauner, and Peter K Sorger. Informatics and quantitative analysis in biological imaging. *Science (New York, N.Y.)*, 300(5616):100–2, April 2003.

[116] Jim Swoger, Francesco Pampaloni, and Ernst H K Stelzer. Light-sheet-based fluorescence microscopy for three-dimensional imaging of biological samples. *Cold Spring Harbor protocols*, 2014(1):1–8, January 2014.

[117] Y. W Teh, M Jordan, Matthew Beal, and David Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1556–1581, 2006.

[118] Yee-Whye Teh. Dirichlet Processes: Tutorial and Practical Course, 2007.

[119] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, May 2001.

[120] Johnny M Tkach, Askar Yimit, Anna Y Lee, Michael Riffle, Michael Costanzo, Daniel Jaschob, Jason A Hendry, Jiongwen Ou, Jason Moffat, Charles Boone, Trisha N Davis, Corey Nislow, and Grant W Brown. Dissecting DNA damage response pathways by analysing protein localization and abundance changes during DNA replication stress. *Nature cell biology*, 14(9):966–76, September 2012.

[121] Pavel Tomancak, Benjamin P Berman, Amy Beaton, Richard Weiszmann, Elaine Kwan, Volker Hartenstein, Susan E Celniker, and Gerald M Rubin. Global analysis of patterns of gene expression during Drosophila embryogenesis. *Genome biology*, 8(7):R145, January 2007.

[122] A H Tong, M Evangelista, A B Parsons, H Xu, G D Bader, N Pagé, M Robinson, S Raghibizadeh, C W Hogue, H Bussey, B Andrews, M Tyers, and C Boone. Systematic genetic analysis with ordered arrays of yeast deletion mutants. *Science (New York, N.Y.)*, 294(5550):2364–8, December 2001.

[123] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: a large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–70, November 2008.

[124] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 1096–1103, New York, New York, USA, July 2008. ACM Press.

[125] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[126] Franco J Vizeacoumar, Nydia van Dyk, Frederick S Vizeacoumar, Vincent Cheung, Jingjing Li, Yaroslav Sydorskyy, Nicolle Case, Zhijian Li, Alessandro Datti, Corey Nislow, Brian Raught, Zhaolei Zhang, Brendan Frey, Kerry Bloom, Charles Boone, and Brenda J Andrews. Integrating high-throughput genetic interaction mapping and high-content screening to explore yeast spindle morphogenesis. *The Journal of cell biology*, 188(1):69–81, January 2010.

[127] C Wählby, I-M Sintorn, F Erlandsson, G Borgefors, and E Bengtsson. Combining intensity, edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections. *Journal of microscopy*, 215(Pt 1):67–76, July 2004.

[128] Martin J. Wainwright and Michael I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(12):1–305, 2007.

[129] David Warde-Farley, Sylva L Donaldson, Ovi Comes, Khalid Zuberi, Rashad Badrawi, Pauline Chao, Max Franz, Chris Grouios, Farzana Kazi, Christian Tannus Lopes, Anson Maitland, Sara Mostafavi, Jason Montojo, Quentin Shao, George Wright, Gary D Bader, and Quaid Morris. The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function. *Nucleic acids research*, 38(Web Server issue):W214–20, July 2010.

[130] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in Neural Information Processing Systems 18*, 10:1473–1480, 2006.

[131] Zheng Yin, Xiaobo Zhou, Chris Bakal, Fuhai Li, Youxian Sun, Norbert Perrimon, and Stephen T C Wong. Using iterative cluster merging with improved gap statistics to perform online phenotype discovery in the context of high-throughput RNAi screens. *BMC bioinformatics*, 9(1):264, January 2008.

[132] Xiaodong Yu. Derivation of Gibbs Sampling for Finite Gaussian Mixture Model. Technical report, University of Maryland, College Park, 2009.

[133] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. Technical report, New York University, December 2012.

[134] Mu Zhu. Kernels and Ensembles. *The American Statistician*, 62(2):97–109, May 2008.