QUASI-NEWTON METHODS FOR STOCHASTIC OPTIMIZATION AND PROXIMITY-BASED METHODS FOR DISPARATE INFORMATION FUSION

Brent Castle

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the School of Informatics and Computing,
Indiana University.

July 2012
Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree Doctor of Philosophy.

Doctoral Committee

Michael Trosset, Ph.D.
(Principal Advisor)

Dirk Van Gucht, Ph.D.

Mehmet Dalkilic, Ph.D.

July 11, 2012

Guilherme Rocha, Ph.D.
This dissertation is dedicated to the memory of my grandfather, William S. Castle.
Acknowledgements

First, I would like to thank my advisor Michael Trosset whose influence is apparent in every aspect of this dissertation. Michael has been a great mentor, advocate, and friend over the last five years. His interests outside of statistics are vast and intersect many of mine. We have shared many hours sidetracked in enjoyable conversation on topics ranging from heist films to cycling. These conversations will be greatly missed.

I would also like to thank my research committee members Dirk Van Gucht, Mehmet Dalkilic, and Guilherme Rocha for their various contributions to my development throughout graduate school and their invaluable feedback improving the clarity and quality of my research. I would be remiss if I did not also acknowledge Carey Priebe. Carey has played a critical role in guiding my research interests, challenging my ideas, and providing thoughtful feedback.

I am indebted to Dana Fielding for making graduate school a much more enjoyable experience than it would have been without her. Dana is always willing to lend an ear and a hand.

My parents Terry and Karen have been supportive throughout all of my schooling and various careers. Without their confidence, love, and endless support I would not be where I am today. I would like to acknowledge my sister Heather, brother-in-law Neil, and the rest of my extended family for their support over the years. In particular, I would like to acknowledge my grandfather.
William Castle who taught me the importance of enjoying your work, working hard, and staying humble.

A key ingredient in maintaining perspective throughout the PhD process is having great friends both in and out of school. Minh Tang and I have shared many great times traveling to conferences and workshops, digging through book stores, playing table tennis, and enjoying meals across the continent. Colin Smith has provided hours of enjoyable conversation discussing everything from politics to our unrelated research areas to the politics of our research areas. I would also like to acknowledge new friends and old including Todd Windler, Mark Rohrer, Chris and Jean Ross, Colleen Iudice, David Pelissier, and all of the various members of the Wednesday Wellington group for countless fun times over the years.

Lastly, I would like to thank my beautiful wife Kendra. She is my best friend, my biggest supporter, my better half, and my lifelong companion. I could not have made it through graduate school without her love and support.
Preface

This dissertation is written in two parts. Each part describes work done in support of one of two grants that funded my graduate studies at Indiana University, one from the Air Force Office of Scientific Research and one from the Office of Naval Research. Part I concerns the use of quasi-Newton methods for stochastic optimization. This body of work is largely a by-product of the development of QNSTOP, a family of quasi-Newton methods for stochastic optimization. Part II is a collection of four chapters describing four different aspects of disparate information fusion. The common thread throughout these four chapters is the use of pairwise proximities (dissimilarities or similarities) as a common representation of disparate data.

Part I comprises Chapters 1 through 5. Chapter 1 introduces a general formulation for stochastic optimization problems. The focus is primarily on simulation-based parameter estimation, however, the formulation and methodology are more general. We describe two motivating problems: one positioning ambulance bases optimally in a simulated city and one estimating the parameters of a breast cancer tumor model. Chapter 2 surveys stochastic optimization methods and focuses, in particular, on response surface methodology (RSM) and stochastic approximation (SA). Chapter 3 is the heart of Part I. We introduce QNSTOP, a class of quasi-Newton methods for stochastic optimization. We adapt features from deterministic optimization methods and use them to enhance what could be considered a response surface method. Chapter 4 gives a convergence analysis
of QNSTOP methods. Despite the differences between response surface methods and stochastic approximation we are able to adapt convergence theory developed for stochastic approximation to QNSTOP. Chapter 5 gives numerical results to a suite of test problems and to the example problems described in Chapter 1.

Part II comprises Chapters 7 through 10. Each of the four chapters describe a proximity-based approach to inference, classification, or visualization. The chapters each stand on their own, however, as a collective they illustrate the versatility of using pairwise proximities, when available, as an effective intermediate representation of disparate data.

The two parts of this dissertation seemingly cover two unrelated topics, however, we expect that future research will elucidate many areas where they intersect. Recently, for example, the machine learning community has embraced stochastic optimization as a means to handle massive datasets. In this work, we focus on each topic in isolation.
QUASI-NEWTON METHODS FOR STOCHASTIC OPTIMIZATION AND PROXIMITY-BASED
METHODS FOR DISPARATE INFORMATION FUSION

The difficulty of modeling and analyzing complicated data sets and data generating processes poses many difficult computational challenges in machine learning and statistics. We investigate two particular topics in this dissertation: methods for stochastic optimization and methods exploiting disparate data.

We propose a class of quasi-Newton methods for stochastic optimization, i.e., optimization of a function given noisy function evaluations. In particular, we focus attention on the optimization of analytically intractable functions that we estimate pointwise by simulation. These methods adapt ideas from response surface methodology and stochastic approximation and integrate tools from numerical optimization, in particular, secant updates and trust regions. We develop a convergence theory and evaluate performance on simulated and real-world problems.

Massive datasets consisting of disparate representations of objects are ubiquitous. An example of such data is a set of captioned images, that is to say, paired text and images. We propose methodology for statistical inference, classification, and visualization given data from disparate data types. In this work, we use pairwise proximities as a common representation of the data.
2.4 Some Methods from Deterministic Optimization .............................................. 29

3 QNSTOP ............................................................................................................. 34

3.1 Estimating the Gradient .................................................................................. 36
3.2 Updating the Model Hessian Matrix ................................................................. 39
3.3 Step Length Control ....................................................................................... 42
3.4 Updating the Experimental Design Region ....................................................... 46
3.5 Stopping Criteria .............................................................................................. 49
3.6 A Basic QNSTOP Algorithm .......................................................................... 50

4 Convergence Theory .......................................................................................... 52

4.1 Background and Notation ............................................................................... 53
4.2 An Almost Sure Convergence Theorem .......................................................... 55
4.3 Almost Sure Convergence Theory for QNSTOP ............................................. 58

5 Numerical Experiments ..................................................................................... 66

5.1 Comparing Stochastic Optimization Algorithms ........................................... 66
5.2 Interpolation or Regression? .......................................................................... 68
5.3 General Stochastic Optimization – Quasi-Convex Functions ......................... 73
5.4 Simulation Optimization – Ambulance Base Location .................................. 76
5.5 Parameter Estimation – Tumor Recurrence ................................................... 76
Part I

Quasi-Newton Methods for
Stochastic Optimization
1

Stochastic Optimization

Computer simulation is an easy game playable by an unskilled practitioner with access to a package; by selecting suitable variables, relationships and parameter values one can start an epidemic, fight a war or attempt to manage an economy.

– G. J. S. Ross

The general problem of optimization concerns finding the decision variable \( \theta \) in the feasible set \( \Theta \subseteq \mathbb{R}^p \) that minimizes (or maximizes) a function \( f \). We focus on the problem of minimization and compactly write the problem:

\[
\min \quad f(\theta) \quad \quad (1.1)
\]

\[
\text{s.t.} \quad \theta \in \Theta \subseteq \mathbb{R}^p.
\]

When \( f(\theta) \) is known exactly we refer to (1.1) as a deterministic optimization problem. There are many scenarios, however, where \( f(\theta) \) is unknown and must be estimated. This is the problem of stochastic optimization.
1. Stochastic Optimization

Section 1.1 describes a general framework for stochastic optimization that we use throughout Part I of this dissertation. In particular, we focus on problems where the objective is estimated using Monte Carlo simulations. To demonstrate such a problem we describe an application positioning ambulance bases in a city that is modeled by simulation. Section 1.2 describes a parameter estimation problem that necessitates simulations. We describe an example estimating the parameters of a tumor model.

1.1 Introduction

The uncertainty in the stochastic optimization setting is accounted for by random variables. First, we consider the case of additive noise. We assume there is an unknown function $g : \mathbb{R}^p \rightarrow \mathbb{R}$ and we observe $Y_\theta = g(\theta) + \epsilon_\theta$, where $\epsilon_\theta$ is a random variable representing noise and/or measurement error at $\theta$. Assume we are interested in small values of $\mathbb{E}[Y_\theta]$, i.e., we seek to solve

$$
\min \quad \mathbb{E}[Y_\theta] \\
\text{s.t.} \quad \theta \in \Theta \subseteq \mathbb{R}^p.
$$

We refer to (1.2) as the expectation minimization problem. For concreteness, consider the case where $g(\theta) = ||\theta||^2$ given observations of $Y_\theta = g(\theta) + \epsilon_\theta$ where $\epsilon_\theta \sim \text{Normal}(0, 1)$. One approach to gain information about $g$ at $\theta$ is to observe multiple realizations of the random variable $Y_\theta$ and to estimate it with the sample mean. This type of formulation is common, however, it is not sufficiently general to provide an intuitive framework for casting other types of stochastic optimization problems.
1. Stochastic Optimization

Consider a different formulation where the objective function must be estimated. Let

\[ \mathcal{P} = \{ P(\cdot; \theta) : \theta \in \Theta \subseteq \mathbb{R}^p \} \]  

(1.3)

denote a family of probability distributions of real-valued random variables parameterized by \( \theta \). Let \( T \) denote a statistical functional, i.e., a real-valued function defined on a set of distribution functions. Suppose we seek \( P \in \mathcal{P} \) such that \( T(P) \) is small, i.e., we want to solve:

\[
\min \ f(\theta) = T(P(\cdot; \theta)) \\
\text{s.t. } \theta \in \Theta \subseteq \mathbb{R}^p.
\]

(1.4)

If \( P \) is analytically intractable or unknown then we must estimate \( T(P(\cdot; \theta)) \). When this is the case we refer to (1.4) as the \textit{general stochastic optimization problem}.

To obtain information about \( P(\cdot; \theta) \) we further suppose that we can sample from it. Let 

\[ p_1, \ldots, p_n \sim P(\cdot; \theta) \]

be an IID (independent and identically distributed) sample for fixed \( \theta \). We estimate \( P(\cdot; \theta) \) by the empirical distribution function (edf)

\[
\hat{P}_n(z; \theta) = n^{-1} \sum_{i=1}^{n} I(p_i \leq z)
\]

where \( I(\cdot) \) is the indicator function. By the Glivenko-Cantelli Theorem we know that

\[
\sup_{z \in \mathbb{R}} \left| \hat{P}_n(z; \theta) - P(z; \theta) \right| \rightarrow 0
\]
1. Stochastic Optimization

Almost surely. Moreover, an inequality due to Dvoretzky, Kiefer, and Wolfowitz [42] establishes a rate of convergence. Precisely, the Dvoretzky-Kiefer-Wolfowitz (DKW) Inequality states that for any \( d > 0 \) and all \( n = 1, 2, \ldots \)

\[
\mathbb{P} \left( \sup_{z \in \mathbb{R}} \left| \hat{P}_n(z; \theta) - P(z; \theta) \right| > d \right) \leq 2e^{-2nd^2}.
\]

Because \( \hat{P}_n(\cdot; \theta) \) is a good approximation of \( P(\cdot; \theta) \) we expect the plug-in estimator

\[
\hat{f}_n(\theta) = T(\hat{P}_n(\cdot; \theta))
\]

to be a good estimator for \( f(\theta) = T(P(\cdot; \theta)) \). If \( T \) can be written \( T(P(\cdot; \theta)) = \int r(z)dP(z; \theta) \) for some \( r(z) \) then it is called a linear functional. The plug-in estimator for linear functionals is asymptotically normal, i.e.,

\[
\sqrt{n} \left( T(\hat{P}_n(\cdot; \theta)) - T(P(\cdot; \theta)) \right) \overset{d}{\rightarrow} \text{Normal}(0, \sigma^2(\theta))
\]

so long as \( 0 < \int r^2(z)dP(z; \theta) - (\int r(z)dP(z; \theta))^2 = \sigma^2(\theta) < \infty \) [31]. For more general functionals, the Hadamard differentiability of a related functional is used to establish asymptotical normality [31, Section 4.4], however, this is not necessarily a straightforward task.

Note that the expectation minimization problem described above can be cast in this framework. Let \( T \) denote the mean functional, i.e.,

\[
T(P(\cdot; \theta)) = \int_{y_\theta} dP(y_\theta; \theta).
\]

where \( P(\cdot; \theta) \) denotes the distribution of \( Y_\theta \). Let \( \bar{Y}_{\theta,n} \) denote the mean of \( n \) realizations of \( Y_\theta \). The
1. Stochastic Optimization

plug-in estimator is

\[ T(\hat{P}_n(\cdot; \theta)) = \int y_{\theta} d\hat{P}(y_{\theta}; \theta) \]
\[ = \bar{Y}_{\theta,n}. \]

In the expectation minimization context we note that the estimator is unbiased, i.e.,

\[ E \left[ T(\hat{P}_n(\cdot; \theta)) \right] = T(\hat{P}(\cdot; \theta)); \]

however, this is not the case for general \( T \).

For many choices of \( T \), the estimator \( \hat{f}_n(\theta) \) is biased. In some scenarios the bias can be decreased by increased sampling, however, in the context of optimization it is often unnecessary to excessively sample when the iterates are far from a minimizer. The balance between estimating the objective at unexplored \( \theta \in \Theta \) and obtaining better estimates of \( f(\theta) \) at a particular \( \theta \) is known as the exploration-exploitation tradeoff. A fundamental discovery in stochastic optimization, however, is that well-designed algorithms on well-behaved problems implicitly average without the need for excessive exploitation at particular locations [69, 43]. We address this in more detail in Chapter 2.

The following is an example of an expectation minimization problem where the objective function is estimated by Monte Carlo simulation. This ambulance base location problem was originally described by Pasupathy and Henderson [63].

**Example 1.1 Ambulance Base Location Problem**

Consider the placement of \( b \) ambulance bases in a city represented by the region \([0, 1] \times [0, 1]\). Emergency calls arrive according to a Poisson process with rate parameter \( \lambda_a \) and
1. Stochastic Optimization

location following a bivariate joint triangle density with mode at \((o_x, o_y)\), i.e., 

\[ g_{xy}(x, y) = g_x(x)g_y(y) \]

where

\[
\begin{align*}
g_x(x) &= \begin{cases} 
0 & x < 0 \\
\frac{2x}{o_x} & 0 \leq x \leq o_x \\
\frac{2(1-x)}{1-o_x} & o_x < x \leq 1 \\
0 & x > 1
\end{cases} \quad \text{and} \quad g_y(y) &= \begin{cases} 
0 & y < 0 \\
\frac{2y}{o_y} & 0 \leq y \leq o_y \\
\frac{2(1-y)}{1-o_y} & o_y < y \leq 1 \\
0 & y > 1.
\end{cases}
\end{align*}
\]

Each base hosts an ambulance and when an ambulance is at its base it is available for dispatch. The nearest available ambulance will be dispatched to the site of a call. If no ambulance is available, incoming calls wait in a first in, first out (FIFO) queue. A dispatched ambulance travels to the site of a call, spends time at the site, and returns to the base. The time spent at the scene follows a Poisson process with rate parameter \(\lambda_s\). We refer to the response time as the time waiting in the queue plus the one-way travel time of the ambulance to the scene. Ambulances travel at an average speed of \(v\) along city blocks, i.e., they travel from their base at \((x_1, y_1)\) to the site \((x_2, y_2)\) by first traveling to \((x_1, y_2)\), or equivalently \((x_2, y_1)\).

Let \(\theta = (x_1, y_1, x_2, y_2, \ldots, x_b, y_b)^T\) be the 2b-vector of ambulance locations and \(m = (o_x, o_y, b, \lambda_a, \lambda_s, v)^T\) be the parameter of the city model. For reasonable \(m\), this process is an example of a regenerative queueing process, i.e., all ambulances will be simultaneously available for an incoming call infinitely often as time goes to infinity. If this was not the case, queue wait times would tend to infinity. A regeneration period begins when an incoming call is received and all ambulances are available and ends when the subsequent regeneration period begins.
A typical objective for the above problem is to minimize

\[ f(\theta) = \frac{\mathbb{E}[\text{sum of response times in a regeneration period}]}{\mathbb{E}[\text{# of calls in regeneration period}]} , \]

however, the estimator

\[ \hat{f}_n(\theta) = \frac{\text{sum of observed response times in } n \text{ regeneration periods}}{\text{# of calls in } n \text{ regeneration periods}} \]

obtained by sampling \( n \) regeneration periods is biased. See [78, Section 14.2] for a general discussion of objective functions for queueing problems.

In place of the objective and estimator above we could use a related objective with an unbiased estimator. The mean waiting time during a regeneration period is independent of the mean waiting time of previous regeneration periods. Moreover, they are identically distributed for fixed \( \theta \) and \( m \). Let \( W_{\theta,m} \) denote the random variable representing mean waiting times, \( P(\cdot; \theta, m) \) denote its distribution, and \( T \) denote the mean functional. We seek to solve

\[
\min \ f(\theta) = T(P(\cdot; \theta, m)) = \int \omega_{\theta,m} dP(\omega_{\theta,m}; \theta, m) = \mathbb{E}[W_{\theta,m}]
\tag{1.5}
\]

\[
\text{s.t. } \theta \in [0,1]^{2d} .
\]

This optimization problem (1.5) is analytically intractable, however, we can estimate the objective function by Monte Carlo simulation. We estimate \( f(\theta) \) by observing the mean waiting time in \( n \) simulated regeneration periods. Let \( w_i \) denote the mean waiting time in the \( i \)th regeneration period and \( \hat{P}_n(\cdot; \theta, m) \) denote the empirical distribution of the
1. Stochastic Optimization

$n$ simulated mean waiting times. We estimate $f(\theta)$ by

$$\hat{f}_n(\theta) = T(\hat{P}_n(\cdot; \theta, m)) = \int w_{\theta, m} d\hat{P}_n(w_{\theta, m}; \theta, m) = \frac{1}{n} \sum_{i=1}^{n} w_i.$$  

Because it is unknown how long a regeneration period is, this type of objective can be impractical. Experiments in [25], however, use an easily interpreted objective. They simulate the system for a fixed amount of time and compute the mean waiting time amongst the calls serviced in that time period. Numerical experiments for different choices of $o_x, o_y, b, \lambda_a, \lambda_x,$ and $v$ using this objective are described in Section 5.4.

1.2 Simulation-Based Parameter Estimation

A stochastic optimization problem arises naturally in parameter estimation. Let $q_1, q_2, \ldots, q_m$ denote a set of $m$ real-valued independent observations of a phenomenon and $\mathcal{P} = \{ P(\cdot; \theta) : \theta \in \Theta \subseteq \mathbb{R}^p \}$ a family of model distributions. The problem of estimating $\theta$ such that the observations could have plausibly been drawn from $P(\cdot; \theta)$ by using Monte Carlo simulation is called simulation-based parameter estimation.

Early references to simulation-based parameter estimation date back to the 1970s. Hoel and Mitchell [37] proposed an approach for fitting a cellular proliferation model for E. coli to data collected in [46]. Let $q_{(1)} \leq q_{(2)} \leq q_{(m)}$ denote ordered birth times for E. coli. Hoel and Michell postulated a model distribution $P(\cdot; \theta)$ from which they draw the random IID sample $p_1, \ldots, p_m \sim P(\cdot; \theta)$ and posed the objective function

$$f(\theta) = E \left[ \sum_{i=1}^{m} (p_{(i)} - q_{(i)})^2 \right]$$
where \( p(1) \leq \ldots \leq p(m) \) denote the ordered samples and the expectation is taken with respect to the sample. Naturally, they estimated \( f \) by

\[
\hat{f}_m(\theta) = \sum_{i=1}^{m} (p(i) - q(i))^2
\]

for a particular sample \( p(1) \leq \ldots \leq p(m) \).

Ross [70] described a likelihood approach where the likelihood function is approximated using simulated samples and the parameters are updated by maximizing the approximated likelihood given the observed data. Diggle and Gratton [27] gave a more detailed account of the likelihood approach. Consider the family of density functions \( \{\phi(\cdot; \theta) : \theta \in \Theta\} \) associated with a family of distributions \( P = \{P(\cdot; \theta) : \theta \in \Theta\} \). Diggle and Gratton seek to maximize the log-likelihood, or equivalently, minimize

\[
f(\theta) = -\sum_{i=1}^{m} \log \phi(q_i; \theta).
\]

They estimate the objective for a fixed \( \theta \) by sampling \( p_1, p_2, \ldots, p_n \sim P(\cdot; \theta) \) and computing

\[
\hat{f}_n(\theta) = -\sum_{i=1}^{m} \log \hat{\phi}(q_i; \theta)
\]

where \( \hat{\phi}(\cdot; \theta) \) is a kernel density estimate computed from the sample \( p_1, p_2, \ldots, p_n \).

In a series of papers including Atkinson, Bartoszynski, Brown, and Thompson [1] and Thompson, Atkinson, and Brown [83] the authors promoted a similar approach utilizing different criteria in lieu of log-likelihood. In regards to the approach in [27], the authors in [83] stated:
1. Stochastic Optimization

We feel that the density estimation is unnecessary (and it is computationally expensive); in addition, a bad choice of the smoothing parameter in the density estimation can cause instability in the estimation, leading to real time defaults to the user. There are simpler, cheaper, and more easily automated alternatives.

In [83], they considered three objective functions using observations partitioned into bins. They argue that for a good choice of \( \theta \) the proportion of simulated samples in each bin should approximate the proportion of observed samples in the corresponding bin. One particular objective in [83] is Pearson’s chi-square goodness-of-fit measure with \( k \) bins

\[
f(\theta) = \sum_{j=1}^{k} \left( \frac{(s_j(\theta) - o_j)^2}{o_j} \right)
\]

where \( o_j \) denotes the proportion of observed samples that fall into the \( j \)th bin and \( s_j(\theta) \) denotes the proportion of simulated samples that fall into the \( j \)th bin as the simulation sample size goes to infinity. The objective is then estimated by fixing a sample size \( n \) and plugging in

\[
f_n(\theta) = \sum_{j=1}^{k} \left( \frac{\hat{s}_{nj}(\theta) - o_j)^2}{o_j} \right)
\]

where \( \hat{s}_{nj}(\theta) \) is the proportion of the \( n \) simulated samples that fall into the \( j \)th bin. They refer to the process of estimating parameters of a stochastic process by comparing simulated samples with observed samples as SIMEST.

We adapt an approach from robust parameter estimation using a procedure similar in spirit to those above. Consider the problem of minimum distance estimation. Given observed data \( q_1, q_2, \ldots, q_m \), corresponding empirical distribution \( \hat{Q}_m \), and the family of distributions \( \mathcal{P} = \ldots \)
1. Stochastic Optimization

\{ P(\cdot; \theta) : \theta \in \Theta \subseteq \mathbb{R}^p \} the minimum distance estimate \( \hat{\theta} \) satisfies

\[
\Delta \left( P(\cdot; \hat{\theta}), \hat{Q}_m \right) = \inf \left\{ \Delta \left( P(\cdot; \theta), \hat{Q}_m \right) : \theta \in \Theta \subseteq \mathbb{R}^p \right\}
\]

where \( \Delta \) is a measure of discrepancy between two distributions. Common choices for \( \Delta \) for minimum distance estimation include the Kolmogorov-Smirnov test statistic:

\[
\Delta_{\text{ks}} \left( P(\cdot; \theta), \hat{Q}_m(\cdot) \right) = \sup_{z \in \mathbb{R}} \left| P(z; \theta) - \hat{Q}_m(z) \right|
\]

and the Cramér-von Mises test statistic:

\[
\Delta_{\text{cvm}} \left( P(\cdot; \theta), \hat{Q}_m(\cdot) \right) = \int_{-\infty}^{\infty} \left( P(z; \theta) - \hat{Q}_m(z) \right)^2 dP(z; \theta).
\]

We pose the problem of finding the minimum distance estimator in the general stochastic optimization framework as follows:

\[
\min_{\theta} f(\theta) = T(P(\cdot; \theta)) = \Delta(P(\cdot; \theta), \hat{Q}_m) \quad (1.6)
\]

s.t. \( \theta \in \Theta \subseteq \mathbb{R}^p. \)

If \( P(\cdot; \theta) \) is analytically intractable or unknown, we estimate \( f(\theta) \) by the plug-in estimator:

\[
\hat{f}_n(\theta) = \Delta \left( \hat{P}_n(\cdot; \theta), \hat{Q}_m \right)
\]

where \( \hat{P}_n(\cdot; \theta) \) is the empirical distribution function for \( n \) IID samples from \( P(\cdot; \theta) \). In the case of
1. **Stochastic Optimization**

the Kolmogorov-Smirnov test statistic this means the plug-in estimator is

\[
\hat{f}_n(\theta) = \Delta_{\text{ks}} \left( \hat{P}_n(\cdot; \theta), \hat{Q}_m(\cdot) \right) \\
= \sup_{z \in \mathbb{R}} \left| \hat{P}_n(z; \theta) - \hat{Q}_m(z) \right|
\]

Similarly, the plug-in estimator in the Cramér-von Mises case is

\[
\hat{f}_n(\theta) = \Delta_{\text{cvm}} \left( \hat{P}_n(\cdot; \theta), \hat{Q}_m(\cdot) \right) \\
= \int_{-\infty}^{\infty} \left( \hat{P}_n(z; \theta) - \hat{Q}_m(z) \right)^2 d\hat{P}_n(z) \\
= \frac{1}{n} \sum_{i=1}^{n} \left( \hat{P}_n(p_i; \theta) - \hat{Q}_m(p_i) \right)^2.
\]

The distributions for the Kolmogorov-Smirnov test statistic and the Cramér-von Mises test statistic are not asymptotically normal like the linear functionals discussed above. Below, however, we demonstrate consistency of the Kolmogorov-Smirnov test statistic and use the proof to motivate a heuristic for choosing sample size. The Kolmogorov-Smirnov test statistic

\[
\hat{f}_n(\theta) = \Delta_{\text{ks}} \left( \hat{P}_n(\cdot; \theta), \hat{Q}_m(\cdot) \right) = \sup_{z} \left| \hat{P}_n(z; \theta) - \hat{Q}_m(z) \right|
\]

is used to estimate the objective function

\[
f(\theta) = \Delta_{\text{ks}} \left( P(\cdot; \theta), \hat{Q}_m(\cdot) \right) = \sup_{z} \left| P(z; \theta) - \hat{Q}_m(z) \right|.
\]
First, we establish
\[
\sup_z \left| P(z; \theta) - \hat{Q}_m(z; \theta) \right| = \sup_z \left| P(z; \theta) - \hat{P}_n(z; \theta) + \hat{P}_n(z; \theta) - \hat{Q}_m(z; \theta) \right| \\
\leq \sup_z \left| P(z; \theta) - \hat{P}_n(z; \theta) \right| + \sup_z \left| \hat{P}_n(z; \theta) - \hat{Q}_m(z; \theta) \right| \tag{1.7}
\]

and similarly
\[
\sup_z \left| \hat{P}_n(z; \theta) - \hat{Q}_m(z; \theta) \right| = \sup_z \left| \hat{P}_n(z; \theta) - P(z; \theta) + P(z; \theta) - \hat{Q}_m(z; \theta) \right| \\
\leq \sup_z \left| \hat{P}_n(z; \theta) - P(z; \theta) \right| + \sup_z \left| P(z; \theta) - \hat{Q}_m(z; \theta) \right|. \tag{1.8}
\]

By (1.7) and the DKW Inequality we have
\[
P\left( \left| f(\theta) - \hat{f}_n(\theta) \right| > d \right) = P\left( \sup_z \left| P(z; \theta) - \hat{Q}_m(z; \theta) \right| - \sup_z \left| \hat{P}_n(z; \theta) - \hat{Q}_m(z; \theta) \right| > d \right) \\
\leq 2e^{-2nd^2}. \tag{1.9}
\]

Similarly, by (1.8) and the DKW Inequality we have
\[
P\left( \left| \hat{f}_n(\theta) - f(\theta) \right| > d \right) = P\left( \sup_z \left| \hat{P}_n(z; \theta) - \hat{Q}_m(z; \theta) \right| - \sup_z \left| P(z; \theta) - \hat{Q}_m(z; \theta) \right| > d \right) \\
\leq 2e^{-2nd^2}. \tag{1.10}
\]

Combining (1.9) and (1.10) we obtain
\[
P\left( \left| \hat{f}_n(\theta) - f(\theta) \right| > d \right) \leq 4e^{-2nd^2}; \tag{1.11}
\]

hence, \( \hat{f}_n(\theta) \xrightarrow{P} f(\theta) \) as \( n \to \infty. \)
Moreover, (1.11) can guide the choice of the sample size $n$. With probability at least $1 - \delta$ for some $\delta \in (0, 1)$, the inequality

$$\left| \hat{f}_n(\theta) - f(\theta) \right| \leq d$$

holds for $n \geq \frac{1}{2d^2}(\log(4) - \log(\delta))$. A number of additional factors such as the computational expense of the sampling process must be considered when choosing an appropriate sample size, however, for some choice of $d$ and $\delta$ the above gives a reasonable heuristic.

Example 1.2 describes a tumor recurrence problem from [1] posed using a minimum distance estimation formulation.

**Example 1.2 Breast Cancer Tumor Model Problem**

Atkinson, Bartoszynski, Brown, and Thompson [1] modeled tumor recurrence, i.e., the time between detection of a primary and secondary tumor, by the following axioms:

1. Tumors originate from a single cell and grow exponentially at rate $\theta_1$.
2. Occurrence of systemic tumors is a Poisson process with rate $\theta_2$.
3. Detection of tumor $j$ is a nonhomogeneous Poisson process with rate $\theta_3 Y_j(t)$, where $Y_j(t)$ is the size of tumor $j$ at time $t$.
4. Until the removal of the primary tumor, metastasis is a nonhomogeneous Poisson process with rate $\theta_4 Y_0(t)$.

We seek the model parameter $\theta \in \Theta = \{(\theta_1, \theta_2, \theta_3, \theta_4)^T : \theta_i > 0, j = 1, \ldots, 4\}$ such that the model described by the above axioms is a reasonable model for clinical observation data. Let $P(\cdot;\theta)$ denote the distribution of simulated tumor recurrence times for a particular model parameter and $\mathcal{P} = \{P(\cdot;\theta) \in \Theta\}$ the set of feasible distributions. $P(\cdot;\theta)$ is intractable, but easily sampled by stochastic simulation [87]. Let $p_1, \ldots, p_n \sim P(\cdot;\theta)$ denote the simulated times generated by:
1. Stochastic Optimization

Repeat until \( p_i > \theta \)

Generate \( U_1, U_2, U_3, U_4 \sim \text{Unif}(0, 1) \)

\( \text{Detect1} \leftarrow \log(1 - (\theta_1/\theta_4) \log U_1)/\theta_1 \)

\( \text{Metastasis} \leftarrow \log(1 - (\theta_1/\theta_4) \log U_2)/\theta_1 \)

\( \text{NewSystemic} \leftarrow (- \log U_3)/\theta_2 \)

If Metastasis > Detect1 then

\( \text{Second} \leftarrow \text{NewSystemic} \)

Else

\( \text{Second} \leftarrow \min(\text{Metastasis}, \text{NewSystemic}) \)

\( \text{Detect2} \leftarrow \log(1 - (\theta_1/\theta_3) \log U_4)/\theta_1 \)

\( p_i \leftarrow \text{Second} + \text{Detect2} - \text{Detect1} \)

Observations \( q_1, \ldots, q_{116} \) were recorded for breast cancer patients at the Curie-Sklodowska Cancer Institute in Warsaw. Let \( \hat{Q}_m \) denote the empirical distribution of these times.

We assume that the observations are independent and identically distributed from some distribution \( Q \) which we approximate by \( \hat{Q}_m \). It is safely assumed that the model above oversimplifies the tumor growth process, i.e., it is likely that \( Q \notin \mathcal{P} \). In spite of this shortcoming, we seek \( \theta \) such that \( P(\cdot; \theta) \) is, according to some measure, the \( P \in \mathcal{P} \) nearest \( \hat{Q}_m \). To do so, we use the Kolmogorov-Smirnov test statistic or the Cramér-von Mises test statistic as described above. Numerical experiments using each objective are described in Section 5.5.

Thompson [82, Section 5.4.2] provided compelling motivation for the simulation approach used for Example 1.2. The derivation of a quadrature approximation to a simplified likelihood function for the above problem took 1.5 man-years of work. However, by using simulations in the way
described, a change in the model can be easily accommodated so long as $P(\cdot; \theta)$ is easily sampled.

In the last decade and a half, the ideas proposed in [37, 70, 1, 27, 83] have seen a resurgence in the research of diseases. So-called Microsimulation Models (MSMs) have been developed for a number of diseases. An MSM is a model for simulating observable outcomes for individuals in a target population. The parameters of the MSM can be calibrated, or optimized, so that the outcomes of the simulations in aggregate resemble the respective outcomes in a target population. Ness et al. [59], Loeve et al. [54], Salomon et al. [72], Chia et al. [13], and Rutter et al. [71] have proposed similar objective functions and simulation procedures for disease MSMs.
Methods for Stochastic Optimization

In the previous chapter, we formulated the general stochastic optimization problem. Standard gradient-based numerical optimization algorithms for minimizing smooth nonlinear functions are not applicable in the stochastic setting due to the absence of gradient information. However, many algorithms developed for stochastic optimization have successfully adapted ideas from their counterparts in the deterministic setting.

In this chapter, we describe methods for stochastic optimization that generalize the steepest descent and quasi-Newton methods for numerical optimization. The purpose of this chapter is to lay a foundation for describing QNSTOP in Chapter 3. We begin by introducing two of the first approaches to stochastic optimization: Stochastic Approximation (SA) in Section 2.1 and Response Surface Methodology (RSM) in Section 2.2. In Section 2.3, we describe recently developed stochastic optimization algorithms with features resembling those in QNSTOP. In Section 2.4, we briefly describe related methods for deterministic optimization.
2. Methods for Stochastic Optimization

2.1 Stochastic Approximation

The history of stochastic approximation begins with an influential paper by Robbins and Monro [69] in 1951 describing an approach for finding the root of a function using only noisy observations. Let $F$ be a real-valued, continuously differentiable function of the real variable $\theta$ with derivative $F'$. If $F$ and $F'$ are known, Newton’s method converges to the root $\theta$ under mild conditions (see [26] for details). An iteration of Newton’s method is defined by

$$
\theta_{k+1} = \theta_k - \frac{1}{F'(\theta_k)} F(\theta_k).
$$

Suppose that $F$ is strictly increasing in a neighborhood of $\theta$. Then, for small $a > 0$ the iterates of the recursion

$$
\theta_{k+1} = \theta_k - a F(\theta_k)
$$

converge to the root $\theta$ provided the initial estimate $\theta_0$ is close enough to $\theta$. Further suppose the observations of $F$ are noisy, i.e., suppose we observe $Y_{\theta_k} = F(\theta_k) + \epsilon_{\theta_k}$ where $\epsilon_{\theta_k}$ is a random variable representing observation noise. For the case $E[Y_{\theta_k}] = F(\theta_k)$, Robbins and Monro [69] proposed the recursion

$$
\theta_{k+1} = \theta_k - a_k Y_{\theta_k}
$$

where the iterates $\theta_k$ are random variables. They established that $\theta_k \to \theta$ in mean square (m.s.) (hence, in probability) if $a_k > 0, a_k \to 0$ as $k \to \infty, \sum_k a_k = \infty$, and $\sum_k a_k^2 < \infty$. The prototypical choice is $a_k = 1/(1 + k)$.

In 1952, Kiefer and Wolfowitz [43] proposed an adaptation of the Robbins-Monro procedure for stochastic optimization. Kiefer and Wolfowitz [43] studied the problem of maximization. Due to convention, we consider the problem of minimization. Let $f$ be a real-valued twice continuously
2. Methods for Stochastic Optimization

differentiable function of the real variable \( \theta \) bounded from below with derivative \( f' \) and minimum \( \theta^* \). Corresponding to the root finding context, Newton’s method can be applied to minimize the function \( f \) by solving for \( f'(\theta) = 0 \), or finding the root of \( f'(\theta) \). Hence, we minimize \( f \) via the iterative update

\[
\theta_{k+1} = \theta_k - [f''(\theta_k)]^{-1} f'(\theta_k).
\]

As in the root-finding context, one can minimize via the update

\[
\theta_{k+1} = \theta_k - a_k f'(\theta_k)
\]

where \( a_k \) is often chosen at each iteration by a line search. This recursion, as well as its multivariate extension, is known as the method of steepest descent. Suppose the observations of \( f \) are noisy, i.e., we observe \( Y_{\theta_k} = f(\theta_k) + \epsilon_{\theta_k} \). Kiefer and Wolfowitz [43] proposed estimating \( f'(\theta_k) \) with the central differencing estimator

\[
\overline{g}_k = \frac{Y_{\theta_k+c_k} - Y_{\theta_k-c_k}}{2c_k}
\]

where \( c_k > 0 \). [43] establishes conditions for \( \theta_k \to \theta^* \) in m.s. (hence, in probability) for the algorithm

\[
\theta_{k+1} = \theta_k - a_k \overline{g}_k
\]

where \( a_k > 0, c_k > 0, c_k \to 0, \sum_k a_k = \infty, \sum_k a_k c_k < \infty, \text{and} \sum_k a_k^2 / c_k^2 < \infty \). The sequences \( \{a_k\} \) and \( \{c_k\} \) are often collectively referred to as the gain sequences. The prototypical choices for the gain sequences are \( a_k = 1/(1 + k) \) and \( c_k = 1/(1 + k)^{1/3} \).

Robbins and Monro [69] and Kiefer and Wolfowitz [43] only considered root-finding and optimization of functions of one real variable. Regardless, the natural extensions to a multivariate
context are often called Robbins-Monro and Kiefer-Wolfowitz algorithms respectively. Blum [4] extended both methods to the multivariate case and established sufficient conditions for almost sure (a.s.) convergence. In [4], the author considered the Kiefer-Wolfowitz algorithm with a forward differencing estimate of the gradient.

For the remainder of this chapter and the chapters that follow we focus exclusively on the multivariate minimization problem. Let \( f : \mathbb{R}^p \mapsto \mathbb{R} \) be a real-valued function of the vector \( \theta \). As before, we observe the noisy observations \( Y_{\theta_k} = f(\theta_k) + \epsilon_{\theta_k} \). The multivariate Kiefer-Wolfowitz algorithm estimates the entries in the gradient vector by the central differencing estimator of the respective partial derivatives, i.e.,

\[
g_k = \begin{bmatrix}
\frac{Y_{\theta_k + c_k e_1} - Y_{\theta_k - c_k e_1}}{2c_k} \\
\vdots \\
\frac{Y_{\theta_k + c_k e_p} - Y_{\theta_k - c_k e_p}}{2c_k}
\end{bmatrix}
\]

where \( e_i \) denotes the unit vector in the \( i \)th coordinate direction. The algorithm proceeds by iterating

\[
\theta_{k+1} = \theta_k - a_k g_k
\]

where the sequences \( \{a_k\} \) and \( \{c_k\} \) satisfy the conditions described in the univariate case.

In the six decades since the introduction of stochastic approximation methods, a number of significant advances have been made to improve the performance of these algorithms in practice. Simultaneous Perturbation Stochastic Approximation (SPSA) [77, 78], in particular, has received significant attention in the last two decades. Whereas the Kiefer-Wolfowitz algorithm requires \( 2p \) function evaluations per iteration, SPSA requires only two regardless of the dimension. SPSA uses
2. Methods for Stochastic Optimization

the recursion (2.1), but instead estimates the gradient by

\[ g_k = \left( \frac{Y_{\theta_k+c_k d_k} - Y_{\theta_k-c_k d_k}}{2c_k} \right) d_k \]  \hspace{1cm} (2.2)

where \( d_k = (d_{k1}, d_{k2}, \ldots, d_{kp})^T \) is often chosen by independently setting each \( d_{ki} \) to +1 or −1 with probability 0.5. Despite significant attention since its proposal in 1992, Kushner and Yin [48] asserted that this is an old idea. In 1978, Kushner and Clark [47] established convergence for a “random directions” approach using the gradient estimator (2.2) with \( d_k \) sampled from the unit sphere. [48, Chapter 10, Section 7] argued that there is essentially no difference in performance between SPSA and the approach proposed in [47]. Moreover, in a comparison to the Kiefer-Wolfowitz algorithm, the section concludes: “the random directions method might be superior when the number of significant parameters is much smaller than the dimension”. Recent surveys of stochastic approximation include books by Spall [78] and Kushner and Yin [48].

2.2 Response Surface Methodology

Like Stochastic Approximation, the seminal paper in Response Surface Methodology (RSM) was also published in 1951. Box and Wilson [8] proposed RSM as a method for finding the optimal operating conditions in an industrial setting, e.g., maximizing yield, using a sequence of designed experiments. In RSM literature, \( \mathbb{E}[Y_{\theta}] \) is referred, often implicitly, as the response surface. The goal is to find the \( \theta \) that maximizes the mean response. RSM models the response locally by fitting a linear or quadratic model using a designed regression experiment. As described in Fu [32], a common two phase approach is:

Phase I: A regression experiment is designed in a region of interest around the current iterate. A
2. Methods for Stochastic Optimization

linear model of the response is constructed by the method of least squares. The subsequent iterate is obtained by stepping in the direction of the negative gradient of the linear model. This phase is then iterated until the magnitude of first-order effects becomes small, i.e., until the norm of the gradient estimate is nearly zero.

Phase II: A second-order experimental design is constructed in a region of interest and a quadratic model is fit by the method of least squares. The minimizer of the quadratic model is often taken as the best estimate of the optimal operating conditions.

Much of the literature on response surface methodology has focused on experimental design. Surveys include books by Box and Draper [6] and Myers and Montgomery [58].

2.2.1 A Connection Between RSM and SA

There is an intimate connection between Phase I of RSM and the Kiefer-Wolfowitz SA algorithm described in Section 2.1. We have not seen this connection explicitly mentioned in the literature, but expect that it is likely known amongst experts in the field. In iteration $k$ of the Kiefer-Wolfowitz algorithm, observations are taken at each of $N = 2p$ design sites centered at the fixed location $\theta_k$.

We define the $N$ design sites

$$x_1 = \theta_k + c_k e_1$$
$$x_2 = \theta_k - c_k e_1$$
$$\vdots$$
$$x_{N-1} = \theta_k + c_k e_p$$
$$x_N = \theta_k - c_k e_p$$
2. Methods for Stochastic Optimization

and refer to this as a central differencing design. For notational convenience we define the \( N \times p \) design matrix

\[
X = \begin{bmatrix}
x_1^T \\
\vdots \\
x_N^T
\end{bmatrix}.
\]

Let \( y_i \) denote an observation of the random variable \( Y_{x_i} \) and let \( Y = (y_1, \ldots, y_N)^T \) denote the vector of responses. Phase I of RSM fits the linear model \( y_i = \beta_0 + x_i^T \beta + e_i \) where \( e_i \) accounts for the lack of fit to a linear model. The least squares estimate of \((\beta_0, \beta)\) minimizes

\[
(Y - \beta_0 1_N - X\beta)^T (Y - \beta_0 1_N - X\beta)
\]

where \( 1_N \) is the \( N \)-vector of ones. RSM is only concerned with the gradient of the linear model, i.e., we seek to compute an optimal \( \beta \). Let

\[
\hat{X} = \begin{bmatrix}
(x_1 - \theta_k)^T \\
(x_2 - \theta_k)^T \\
\vdots \\
(x_{N-1} - \theta_k)^T \\
(x_N - \theta_k)^T
\end{bmatrix} = c_k \begin{bmatrix}
c_k e_1 \\
- c_k e_1 \\
\vdots \\
c_k e_p \\
- c_k e_p
\end{bmatrix} = c_k \begin{bmatrix}
1 & 0 & \ldots & 0 & 0 \\
-1 & 0 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & 0 & 1 \\
0 & 0 & \ldots & 0 & -1
\end{bmatrix}.
\]

Then, we can solve for the least squares optimal \( \beta \) by computing

\[
\hat{\beta} = (\hat{X}^T \hat{X})^{-1} \hat{X}^T Y.
\]
Due to the structure of $\hat{X}$ we see that $(\hat{X}^T\hat{X}) = (2c_k^2)I_p$ where $I_p$ is the identity matrix. Hence,

$$\hat{\beta} = \frac{1}{2c_k} \begin{bmatrix} 1 & -1 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 1 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & -1 \end{bmatrix} Y = \frac{1}{2c_k} \begin{bmatrix} y_1 - y_2 \\ \vdots \\ y_{N-1} - y_N \end{bmatrix}$$

which is evidently the same gradient estimator obtained by central differencing. We conclude that Phase I of RSM using central differencing designs is the Kiefer-Wolfowitz algorithm so long as the step length control and finite differencing interval are given by the gain sequences described in Section 2.1.

### 2.2.2 Ridge Analysis

Phase II of RSM constructs a quadratic model of the response surface in a region of interest. If the minimizer of the quadratic model is outside of the region of interest then ridge analysis provides a method for studying minimizers of the quadratic model subject to a restriction on the distance from the center about which the quadratic was constructed. Consider the quadratic model

$$\hat{m}(x) = \hat{f} + \hat{g}^T(x - \theta_k) + \frac{1}{2}(x - \theta_k)^T\hat{H}(x - \theta_k)$$

(2.3)

where $\theta_k$ was the final iterate of Phase I and $\hat{f}$, $\hat{g}$, and $\hat{H}$ have been fit by least squares. Consider the minimization of $\hat{m}(x)$ subject to the constraint $\frac{1}{2}\|x - \theta_k\|^2 \leq \frac{1}{2}\rho^2$ for some $\rho > 0$. If

$$\arg\min \hat{m}(x) = \theta_k - \hat{H}^{-1}\hat{g}$$

(2.4)
satisfies the constraint, then we are done. Otherwise, the constrained minimizer will satisfy $\frac{1}{2} \| x - \theta_k \|^2 = \frac{1}{2} \rho^2$. We introduce the Lagrange multiplier $\mu$, write the Lagrangian

$$L(x, \mu) = \hat{f} + \hat{g}^T (x - \theta_k) + \frac{1}{2} (x - \theta_k)^T \hat{H} (x - \theta_k) + \mu \left( \frac{1}{2} \| x - \theta_k \|^2 - \frac{1}{2} \rho^2 \right)$$

$$= \hat{f} + \hat{g}^T x - \hat{g}^T \theta_k + \frac{1}{2} \left( x^T \hat{H} x - x^T \hat{H} \theta_k - \theta_k^T \hat{H} x + \theta_k^T \hat{H} \theta_k \right)$$

$$+ \mu \left( \frac{1}{2} x^T x - \theta_k^T x + \frac{1}{2} \theta_k^T \theta_k - \frac{1}{2} \rho^2 \right),$$

and differentiate it with respect to $x$

$$\frac{\partial L}{\partial x} = \hat{g} - \hat{H} \theta_k - \mu \theta_k + \hat{H} x + \mu x. \quad (2.5)$$

An $x$ satisfies first-order necessity conditions for optimality when (2.5) is equal to 0. Hence, an optimal $x$ satisfies

$$\left( \hat{H} + \mu I_p \right) x = \left( \hat{H} + \mu I_p \right) \theta_k - \hat{g}.$$ 

Solving for $x$ we note that the $x$ depends on $\mu$ and we write

$$x(\mu) = \theta_k - \left( \hat{H} + \mu I_p \right)^{-1} \hat{g}. \quad (2.6)$$

The constrained minimizers given various choices of $\rho^2$ form an arc dependent upon the Lagrange multiplier $\mu \geq 0$. We revisit this idea in the discussion of trust region methods in Section 2.4.1 and again in Chapter 3 when we introduce QNSTOP.
2. Methods for Stochastic Optimization

2.3 Additional Methods from Stochastic Optimization

In this section, we briefly review three methods for stochastic optimization that share features in common with methods in QNSTOP.

2.3.1 Lawera-Thompson Algorithm

Lawera and Thompson [50] described a response surface method based on the work of Box and Hunter [7]. They use several heuristic methods for mitigating the negative effects of noisy function evaluations. The primary contributions include adaptive experimental designs and a quasi-trust region step length control.

The Lawera-Thompson Algorithm uses Box-Hunter “cube and star points” rotatable designs. In a standardized scaling, Box-Hunter designs locate \(2^p\) design sites at the “cube points” \(\{(\pm 1, \ldots, \pm 1)^T\}\) and \(2^p\) design sites at the “star points” \(\{(\pm 2^{p/4}, 0, \ldots, 0)^T, (0, \pm 2^{p/4}, 0, \ldots, 0)^T, \ldots, (0, \ldots, 0, \pm 2^{p/4})^T\}\). Multiple replications are observed at the center point. Lawera-Thompson rotates the Box-Hunter designs such that variation of the objective at the ends of each axis defined by the star points is roughly equal. The objective is observed several times at each design site and a quadratic model is fit to the mean response by least squares regression. The algorithm uses the \(r^2\) statistic, error sum of squares, and the magnitude of the difference between the maximum and minimum observed responses at the center point to adjust the scale of the subsequent iteration’s design.

To prevent the algorithm from carelessly stepping to a distant minimizer of the quadratic model, the Lawera-Thompson algorithm implements a quasi-trust region constraint in the following manner. Let \(s \in \mathbb{R}^p\) denote the step to the minimizer of the quadratic model in standard units. If \(||s|| \leq 1\) then the step remains unmodified. Otherwise, the smallest terms in \(s\) are set to 0 until the norm of
the modified step is less than 1. Sections 2.4.1 and 3.3 discuss more common trust region constraints in detail.

2.3.2 Noisy UOBYQA

Powell [66] proposed UOBYQA (Unconstrained Optimization BY Quadratic Approximation) for the deterministic setting. UOBYQA constructs quadratic models by interpolation and progresses by minimizing them subject to a trust-region constraint, i.e., within a predetermined neighborhood of the current iterate.

Deng and Ferris [24] proposed three novel modifications to UOBYQA to adapt it to the stochastic setting. They observe the response at each design site multiple times and interpolate the mean responses. A heuristic is used to determine how many observations should be taken at each design site such that the quadratic model and the constrained minimizer are stable. The constrained minimizer of the quadratic model is computed in the same way as in UOBYQA, however, [24] proposed a novel heuristic for choosing whether to update the current iterate with the minimizer or leave it unchanged. They also describe termination criteria specific to the stochastic setting based upon having similar mean responses amongst a large portion of sites on the boundary of the trust region.

2.3.3 STRONG and STRONG-X

Chang, Hong, and Wan [10] and Chang and Wan [11] proposed the STRONG and STRONG-X algorithms, respectively. STRONG assumes normally distributed function evaluation errors and STRONG-X relaxes this assumption to additive errors with bounded variance. Both algorithms adapt the standard two-phase RSM approach and utilize trust regions to control progress. The first phase constructs a linear model fit partially by least squares to multiple observations at design
sites in an appropriate design (e.g., the authors recommend a fractional factorial or factorial design plus the current iterate). A line search is used in the direction of negative gradient within the trust region to choose the subsequent iterate. The second phase constructs a quadratic model by least squares. If sufficient progress is made, the algorithm steps to the Cauchy point, i.e., the minimizer of the quadratic in the direction of steepest descent subject to the trust region constraint. Heuristics were described to determine whether sufficient progress was obtained in each phase.

### 2.4 Some Methods from Deterministic Optimization

Many approaches to deterministic optimization have features that are also applicable in the context of stochastic optimization. In Sections 2.4.1 and 2.4.2 we introduce trust region methods and secant methods respectively. In Section 2.4.3, we briefly describe Implicit Filtering, a model-based approach utilizing secant updates which shares features with methods in QNSTOP.

#### 2.4.1 Trust Region Methods

Newton’s method for unconstrained minimization of $f : \mathbb{R}^p \mapsto \mathbb{R}$ constructs the quadratic model at the current iterate $\theta_k$

$$m_k(p) = f(\theta_k) + \nabla f(\theta_k)^T p + \frac{1}{2} p^T \nabla^2 f(\theta_k) p.$$  

If $\nabla^2 f(\theta_k)$ is positive definite then $m_k$ is minimized by

$$p_k = -[\nabla^2 f(\theta_k)]^{-1}\nabla f(\theta_k)$$
2. Methods for Stochastic Optimization

and Newton’s Method proceeds by

$$\theta_{k+1} = \theta_k + p_k.$$ 

The quadratic model $m_k$ can provide a good model of the function locally, however, it is often a poor global model. Hence, stepping to the unconstrained minimizer of the quadratic is generally not recommended. One approach to alleviate this complication is to restrict the norm of the update. Trust region methods (see Conn, Gould, and Toint [18] for a comprehensive treatment) compute the update by minimizing $m_k$ subject to the constraint $\|p_k\| \leq \rho k$ where $\rho k > 0$ is the trust region radius for iteration $k$. This constrained optimization problem is often referred to as the trust region subproblem. Conceptually, trust region methods attempt to restrict steps to a region in which you trust the quadratic model. For appropriate updating schemes for $\rho k$, trust region methods are globally convergent.

This approach is clearly related to the constrained minimization of a quadratic model described in Section 2.2.2. Despite both methods existing for several decades, this connection was first mentioned in the literature in 2003 in Trosset [85]. In contrast to ridge analysis, trust region methods fix $\rho k$ at each iteration and attempt to find $\mu$ in (2.6) such that $x(\mu)$ solves the optimization problem. This is a difficult optimization itself, but it has been shown that even crude approximations give rise to globally convergent algorithms.

2.4.2 Quasi-Newton and Secant Methods

It is not always desirable or even feasible to compute the exact Hessian matrix. Quasi-Newton methods model the function locally by the quadratic model

$$\hat{m}_k(p) = f(\theta_k) + \nabla f(\theta_k)^T p + \frac{1}{2} p^T \hat{H}_k p$$
2. Methods for Stochastic Optimization

where $\hat{H}_k$ is a modification or an approximation of $\nabla^2 f(\theta_k)$. The search direction is chosen to be the minimizer of the quadratic, i.e.,

$$p_k = -\hat{H}_k^{-1} \nabla f(\theta_k)$$

and the algorithm proceeds with the update

$$\theta_{k+1} = \theta_k + a_k p_k$$

where $a_k$ controls step length. Secant methods provide a particular approximation of the Hessian matrix with favorable computational and modeling benefits. Secant methods compute an approximation $\hat{H}_{k+1}$ for $\nabla^2 f(\theta_{k+1})$ that satisfies the so-called secant equation [26, Chapter 9]:

$$\hat{H}_{k+1}(\theta_{k+1} - \theta_k) = \nabla f(\theta_{k+1}) - \nabla f(\theta_k). \tag{2.7}$$

Some authors refer to updating methods that satisfy the secant equation as quasi-Newton methods, however, we refer to secant methods as the special case of quasi-Newton method that satisfy the secant equation.

Let

$$\hat{m}_{k+1}(p) = f(\theta_{k+1}) + \nabla f(\theta_{k+1})^T p + \frac{1}{2} p^T \hat{H}_{k+1} p.$$

Secant methods have the desirable property that if (2.7) holds, then the gradient of the quadratic
model \( \tilde{m}_{k+1} \) equals the gradient of the function \( f \) at \( \theta_k \) and \( \theta_{k+1} \), i.e.,

\[
\nabla \tilde{m}_{k+1}(0) = \nabla f(\theta_{k+1}),
\]

\[
\nabla \tilde{m}_{k+1}(-a_k p_k) = \nabla f(\theta_{k+1}) - a_k \tilde{H}_{k+1} p_k
\]

\[
= \nabla f(\theta_k).
\]

Many methods for computing an \( \tilde{H}_{k+1} \) that satisfy (2.7) have been proposed. The BFGS update, named after co-discoverers Broyden, Fletcher, Goldfarb, and Shanno, is undoubtedly the most popular method in the context of unconstrained optimization. Let \( s_k = \theta_{k+1} - \theta_k \) and \( \nu_k = \nabla f(\theta_{k+1}) - \nabla f(\theta_k) \). BFGS updates the Hessian matrix by a rank-two update according to

\[
\tilde{H}_{k+1} = \tilde{H}_k - \frac{\tilde{H}_k s_k s_k^T \tilde{H}_k}{s_k^T \tilde{H}_k s_k} + \frac{\nu_k \nu_k^T}{\nu_k^T s_k}.
\]

If \( \tilde{H}_k \) is positive definite and \( \nu_k^T s_k > 0 \), BFGS produces positive definite \( \tilde{H}_{k+1} \). Positive definiteness ensures that the quadratic model is convex and has a unique minimizer. However, if a trust region is used in combination with the secant update then it is unnecessary to maintain convexity of the model. SR1 (Symmetric Rank 1) is the unique symmetric, rank-one update satisfying (2.7), but it does not guarantee positive definiteness of the updated Hessian approximation. SR1 updates by

\[
\tilde{H}_{k+1} = \tilde{H}_k + \frac{(\nu_k - \tilde{H}_k s_k)(\nu_k - \tilde{H}_k s_k)^T}{(\nu_k - \tilde{H}_k s_k)^T s_k}.
\]

Numerical experiments in [16], [17], and [61] demonstrated that SR1 can outperform BFGS when used with trust regions. In Section 3.2, we describe a novel quasi-secant method that adapts SR1 for use with stochastic optimization methods. We describe the method as a quasi-secant method because we add an additional constraint that might preclude satisfying the secant equation.
2. Methods for Stochastic Optimization

2.4.3 Implicit Filtering

Implicit Filtering [33, 41] addresses the bound-constrained deterministic optimization problem when derivatives are not available. The basic algorithm samples a stencil design, estimates a gradient, and steps in the direction of negative gradient with step length established by a line search. To accelerate convergence, a quasi-Newton direction can be used where the model Hessian is computed by BFGS or SR1. The key features include methods for handling constraints and adjusting the stencil sizes.
In this chapter we describe QNSTOP, a class of quasi-Newton methods for solving the general stochastic optimization problem:

\[
\begin{align*}
\min & \quad f(\theta) \\
\text{s.t.} & \quad \theta \in \Theta \subseteq \mathbb{R}^p
\end{align*}
\]

where we must resort to estimates of \( f(\theta) \). Levy, Trosset, and Kincaid [51] introduced a quasi-Newton method for stochastic optimization. They fit linear models by least squares and used BFGS to compute a model Hessian matrix. The authors identified several areas for possible improvement including the use of trust regions. We view QNSTOP as a descendent of this work.

In iteration \( k \), QNSTOP methods compute the gradient vector and Hessian matrix of a quadratic model

\[
\hat{m}_k(p) = f_k + \hat{g}_k^T p + \frac{1}{2} p^T \hat{H}_k p. \tag{3.1}
\]

of the response surface at \( \theta_k \). The gradient vector \( \hat{g}_k \) is a realization of a random vector denoted \( g_k \) because it is computed from estimates of \( f(\theta_k) \), i.e., realizations of a random variable whose
expectation is ideally near \( f(\theta_k) \). Similarly, the Hessian matrix \( \hat{H}_k \) is a realization of a random matrix denoted \( H_k \). In this chapter, we discuss how a single iteration progresses; hence, we focus on what the algorithm does to obtain \( \hat{g}_k \) and \( \hat{H}_k \) and how it uses these quantities. In Chapter 4, we consider the behavior of the algorithm as a stochastic process.

The methods progress by stepping to a trust region constrained minimizer of the quadratic. Because \( f_k \) does not affect the minimizer of the quadratic, the methods progress by minimizing

\[
\hat{q}_k(p) = \hat{g}_k^T p + \frac{1}{2} p^T \hat{H}_k p.
\]

The methods update the estimated minimizer by:

\[
\theta_{k+1} = \theta_k - [\hat{H}_k + \mu_k W_k]^{-1} \hat{g}_k
\]

where \( \mu_k \) is the Lagrange multiplier of the trust region subproblem and \( W_k \) is a scaling matrix. In spirit, QNSTOP methods are quasi-Newton methods despite the fact that \( \hat{g}_k \) is an estimate of the gradient. In the case where \( \Theta \) is a convex subset of \( \mathbb{R}^p \) we can consider algorithms of the form

\[
\theta_{k+1} = \left( \theta_k - [\hat{H}_k + \mu_k W_k]^{-1} \hat{g}_k \right)_\Theta,
\]

where \( (\cdot)_\Theta \) denotes projection onto \( \Theta \), however, we limit our discussion in this chapter to unconstrained algorithms.

Section 3.1 describes experimental designs suitable for estimating the gradient at \( \theta_k \). Section 3.2 describes a quasi-secant update for computing a model Hessian matrix particularly well-suited for stochastic optimization. As in deterministic optimization, a quadratic model may be a poor global model of the response surface. Moreover, particularly noisy observations might cause the quadratic
model to be a poor local model as well. To address this concern we use trust regions to control step length. Section 3.3 implements two different approaches to trust region constraints in this setting. Section 3.4 concerns updating the ellipsoidal design region to be used by the subsequent iteration. Section 3.5 discusses stopping criteria. Section 3.6 provides a basic QNSTOP algorithm using the ideas in Sections 3.1-3.4.

3.1 Estimating the Gradient

Following the RSM philosophy, we design regression experiments in a region of interest containing the current iterate. To accommodate local rescaling we use an ellipsoidal design region centered at the current iterate $\theta_k \in \mathbb{R}^p$. Let

$$W_\gamma \triangleq \{ W \in \mathbb{R}^{p \times p} : W = W^T, \det(W) = 1, \gamma^{-1}I_p \preceq W \preceq \gamma I_p \}$$

for some $\gamma \geq 1$. The elements of the set $W_\gamma$ are valid scaling matrices which control the shape of the ellipsoidal design regions with eccentricity constrained by $\gamma$. The ellipsoidal design regions are defined by

$$E_k(\tau_k) \triangleq \{ x \in \mathbb{R}^p : (x - \theta_k)^T W_k (x - \theta_k) \leq \tau_k^2 \}$$

where $W_k \in W_\gamma$ and the parameter $\tau_k > 0$ controls the size of the ellipsoid.

In each iteration, QNSTOP methods choose a set of $N_k$ design sites $D_k = \{x_{k1}, \ldots, x_{kN_k}\} \subset E_k(\tau_k)$. For notational simplicity we describe the algorithm as if $N_k$ is a constant $N$. The design has to be useful, in some sense, for fitting a linear model to the response surface by regression. The problem of assessing the quality of a design is addressed by Conn, Scheinberg, and Vicente [19, 20]. The authors described designs that are well poised for both interpolation and regression.
In particular, they describe so-called $\Lambda$-poisedness. We describe $\Pi$-poisedness, a related measure that is simpler in the context of our work. Let $\mathbf{x}_k = N^{-1} \sum_{i=1}^{N} x_{ki}$ denote the mean design site and denote the centered design matrix

$$\mathbf{X}_k \triangleq \mathbf{X}(D_k) = \begin{bmatrix} (x_{k1} - \bar{x}_k)^T \\
\vdots\\n(x_{kN} - \bar{x}_k)^T \end{bmatrix}. $$

Let $\Delta_k = 2\gamma_k \sqrt{\gamma}$ denote an upper bound for the length of the longest axis of the ellipse. Denote the scaled and centered design $\hat{D}_k = \{\Delta_k^{-1}(x_{k1} - \bar{x}_k), \ldots, \Delta_k^{-1}(x_{kN} - \bar{x}_k)\}$ and the respective scaled and centered design matrix $\hat{\mathbf{X}}_k = \Delta_k^{-1} \mathbf{X}_k$. The scaling factor $\Delta_k^{-1}$ ensures each $\hat{x}_{ki} \in \hat{D}_k$ is in $B(0, 1)$, the unit ball centered at 0. Let $\sigma_{\min}(\hat{\mathbf{X}}_k)$ denote the smallest singular value of $\hat{\mathbf{X}}_k$.

**Definition 3.1.** A design $D_k = \{x_{k1}, \ldots, x_{kN}\}$ is $\Pi$-poised iff $\sigma_{\min}(\hat{\mathbf{X}}_k) \geq \Pi > 0$.

[20, Chapter 6] describes how to ensure $\Lambda$-poisedness. The same approaches apply for $\Pi$-poised designs as $\Lambda$-poisedness implies $\Pi$-poisedness. Typical designs used in RSM such as fractional factorial, factorial, and simplex designs are $\Pi$-poised. In the context of an optimization algorithm, we desire all designs $\{D_k\}_{k=0}^{\infty}$ to be $\Pi$-poised for some fixed $\Pi > 0$. In Chapter 4, we discuss the importance of $\Pi$-poised designs. In short, these designs give rise to bounds needed to establish convergence.

### 3.1.1 A Generalized Central Differencing Design

We introduced the central differencing designs used by Kiefer-Wolfowitz in Section 2.2.1. In this section, we generalize the central differencing design such that the design sites are at the extremes of the semi-principal axes of the design ellipsoid. Consider the eigendecomposition of
3. QNSTOP

\[ W_k = V_k \Lambda_k V_k^T \]

where

\[ V_k = [v_{k1} \ldots v_{kp}] \quad \text{and} \quad \Lambda_k = \text{diag}(\lambda_{k1}, \ldots, \lambda_{kp}). \]

A generalized central differencing design is the set

\[ D_k \triangleq \{ \theta_k \pm z_{ki} : z_{ki} = \tau_k \lambda^{-1/2}_{ki} v_{ki}, i = 1, \ldots, p \}. \]

The parameter controlling the eccentricity of the design ellipsoids (\( \gamma \)) ensures the eigenvalues of \( W_k \) are bounded uniformly from above; hence, the generalized central differencing designs are II-poised.

3.1.2 Least Squares Estimate of the Gradient

In each iteration of a QNSTOP method we observe the response at each design site in a II-poised design. Let \( Y_k = (y_{k1}, \ldots, y_{kN})^T \) denote the \( N \)-vector of responses. We model the response surface by the linear model

\[ y_{ki} = f_k + x_{ki}^T \hat{g}_k + e_i \]

where \( e_i \) accounts for lack of fit. The least squares estimate of the gradient \( \hat{g}_k \) is obtained by observing the responses and solving

\[ (X_k^T X_k) \hat{g}_k = X_k^T Y_k. \]

In the case of generalized central differencing designs, an estimate can be computed more efficiently. Consider the directional derivative \( f'(\theta_k)(z_{ki}) \), i.e., the derivative of \( f \) at \( \theta_k \) in the direction \( z_{ki} \). We estimate the directional derivative with the random variate \( f'_{ki} \), an observation of

\[ \frac{Y_{\theta_k + z_{ki}} - Y_{\theta_k - z_{ki}}}{2\tau_k \lambda^{-1/2}_{ki}}. \]
and denote the vector $f'_k = \left( f'_{k1}, \ldots, f'_{kp} \right)^T$. To obtain the gradient estimate one can simply compute $f'_k$ and then $\hat{g}_k = V_k f'_k$. Consider the special case $W_k = I_p$ where $V_k = I_p$ and $\Lambda_k = I_p$.

### 3.2 Updating the Model Hessian Matrix

The Hessian matrix $\hat{H}_k$ of the quadratic model is updated using the gradient estimator $\hat{g}_k$. The update is restricted to satisfy

$$-\eta I_p \preceq \hat{H}_k - \hat{H}_{k-1} \preceq \eta I_p$$

for some $\eta \geq 0$. Conceptually, this prevents the quadratic model from changing drastically from one iteration to the next. Moreover, in the stochastic setting, this helps safeguard the quadratic from particularly noisy observations. In Chapter 4, we give theoretical justification for this particular restriction.

In the early development of QNSTOP, the model Hessian matrix was obtained by either fitting the entire quadratic model by linear regression or it was updated by BFGS or SR1 as described in Section 2.4.2 plugging in $\hat{g}_k$ and $\hat{g}_{k-1}$ in place of $\nabla f(\theta_k)$ and $\nabla f(\theta_{k-1})$ respectively. In this section, we describe CSR1 (Constrained Symmetric Rank 1), a novel quasi-secant update. CSR1 finds a symmetric, rank 1 update that satisfies (3.4) and minimally violates the secant equation (2.7) as measured by Euclidean norm, i.e., we solve

$$\min_{\hat{H}_k} \left\| \hat{H}_k (\theta_k - \theta_{k-1}) - (\hat{g}_k - \hat{g}_{k-1}) \right\|^2$$

s.t.

$$\hat{H}_k = \hat{H}_k^T$$

$$\text{rank}(\hat{H}_k - \hat{H}_{k-1}) = 1$$

$$-\eta I_p \preceq \hat{H}_k - \hat{H}_{k-1} \preceq \eta I_p.$$
Let $\hat{H}^*_k$ denote a solution to (3.5). The standard SR1 (symmetric, rank 1) secant update computes the unique symmetric and rank 1 update matrix that satisfies the secant equation, i.e., the objective (3.5a) is exactly zero and the updated Hessian matrix satisfies the constraints (3.5b) and (3.5c). Let $\hat{H}^*_k$ denote the SR1 updated model Hessian matrix. If $-\eta I_{p} \preceq \hat{H}^*_k - \hat{H}_{k-1} \preceq \eta I_{p}$, then $\hat{H}^*_k = \hat{H}^*_k$ is the unique solution to the above optimization problem. Otherwise, denote the optimal update matrix $U^*_k = \hat{H}^*_k - \hat{H}_{k-1}$. The optimal update matrix is rank 1 and symmetric so we can decompose it as $U^*_k = \lambda^*_k v^*_k v^*_k^T$ where $\|U^*_k\| = |\lambda^*_k|$ and $\|v^*_k\| = 1$. If $\hat{H}^*_k$ does not satisfy (3.5d) then either $\lambda^*_k = -\eta$ or $\lambda^*_k = \eta$. Let

\[
U_k = \lambda^*_k v^*_k v^*_k^T, \\

s_{k-1} = \theta_k - \theta_{k-1}, \\

\nu_{k-1} = \hat{g}_k - \hat{g}_{k-1}, \text{ and} \\

\zeta_{k-1} = \nu_{k-1} - \hat{H}_{k-1} s_{k-1}.
\]

Rewrite (3.5a)

\[
\left\| \hat{H}_k (\theta_k - \theta_{k-1}) - (\hat{g}_k - \hat{g}_{k-1}) \right\|^2 = \left\| (\hat{H}_{k-1} + U_k) s_{k-1} - \nu_{k-1} \right\|^2 \\
= \left\| U_k s_{k-1} - (\nu_{k-1} - \hat{H}_k s_{k-1}) \right\|^2 \\
= \left\| \lambda^*_k v^*_k v^*_k^T s_{k-1} - \zeta_{k-1} \right\|^2.
\]

Solving (3.5) is equivalent to minimizing

\[
h(\lambda, v) = \left\| \lambda^*_k v^*_k v^*_k^T s_{k-1} - \zeta_{k-1} \right\|^2 \quad (3.6)
\]

\[
= \lambda^2 s_{k-1}^T v^T v^T v^T s_{k-1} - 2\lambda \zeta_{k-1}^T v^T s_{k-1} + \zeta_{k-1}^T \zeta_{k-1}
\]
subject to the constraints $\|v\| = 1$ and $\lambda \in \{-\eta, \eta\}$. Because $v^Tv = 1$ at the solution and $\zeta_{k-1}$ is constant, the constrained minimizer of $h(\lambda, v)$ is also the minimizer of \( h^*(\lambda, v) = \lambda^2 s_{k-1}^T v v^T s_{k-1} - 2\lambda \zeta_{k-1}^T v v^T s_{k-1} \) subject to $\|v\| = 1$ and $\lambda \in \{-\eta, \eta\}$. If $\lambda_* = \eta$, then the constrained minimizer of (3.7) minimizes the Rayleigh quotient \( h^*_+(v) = \frac{v^T(\eta^2 s_{k-1}^T s_{k-1} - \eta s_{k-1}^T \zeta_{k-1}^T - \eta \zeta_{k-1}^T s_{k-1}^T)v}{v^Tv} \).

The Rayleigh quotient $h^*_+$ is minimized by an eigenvector associated with the smallest eigenvalue of $(\eta^2 s_{k-1}^T s_{k-1} - \eta s_{k-1}^T \zeta_{k-1}^T - \eta \zeta_{k-1}^T s_{k-1}^T)$. If unique, we denote this eigenvector $v_+$. Otherwise, we select one arbitrarily and denote it by $v_+$. Similarly, if $\lambda_* = -\eta$, we minimize \( h^*_-(v) = \frac{v^T(\eta^2 s_{k-1}^T s_{k-1} + \eta s_{k-1}^T \zeta_{k-1}^T + \eta \zeta_{k-1}^T s_{k-1}^T)v}{v^Tv} \)

and denote a minimizer by $v_-$. Let $S = \{(\eta, v_+), (-\eta, v_-)\}$ denote the set of the two possible minimizers of (3.6) and let \( (\lambda_*, v_*) = \operatorname{argmin}_{(\lambda, v) \in S} h(\lambda, v) \).

Then $\hat{H}_k = \hat{H}_{k-1} + \lambda_* v_* v_*^T$ is the optimal solution of (3.5).

One difficulty with SR1, even in the deterministic setting, is that when $(\nu_{k-1} - \hat{H}_{k-1} s_{k-1})^T s_{k-1}$ is nearly or equal to zero the update should not or cannot be performed. Typically, one way to
prevent this is to skip the update to the Hessian approximation if

$$\frac{|(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})^T s_{k-1}|}{\|\nu_{k-1} - \hat{H}_{k-1}s_{k-1}\| \|s_{k-1}\|}$$

is small, e.g., less than $10^{-8}$ [60, Chapter 8]. The CSR1 update precludes the need to check such a condition. The CSR1 algorithm is written succinctly:

Algorithm 3.1. CSR1

1. Let $\nu_{k-1} = \hat{g}_k - \hat{g}_{k-1}$ and $s_{k-1} = \theta_k - \theta_{k-1}$.

2. If $(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})^T s_{k-1} > 0$, compute the SR1 update matrix

$$U^s_k = \frac{(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})^T}{(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})^T s_{k-1}}.$$

3. If $U^s_k$ is computed in step 2 and $-\eta I_p \preceq U^s_k \preceq \eta I_p$, then set $U^*_k = U^s_k$. Otherwise, compute $v_+$ and $v_-$. If $h(\eta, v_+) \leq h(-\eta, v_-)$, then set $U^*_k = \eta v_+ v_+^T$. Else, set $U^*_k = -\eta v_- v_-^T$.

4. Compute $\hat{H}^*_k = \hat{H}_{k-1} + U^*_k$.

3.3 Step Length Control

In Sections 3.1 and 3.2, we discuss methods for computing the gradient and Hessian of the quadratic model (3.2). In this section we describe two approaches to enforcing a trust region constraint to the minimization of (3.2).
3. QNSTOP

3.3.1 Explicit Trust Region Radii

In analogy with trust region methodology in the deterministic setting (see Section 2.4.1) we can fix a trust region radius $\rho_k$ in each iteration and solve

$$\begin{align*}
\min_{p} \quad & \hat{g}_k(p) = \hat{g}_k^T p + \frac{1}{2} p^T \hat{H}_k p \\
\text{s.t.} \quad & ||p|| \leq \rho_k.
\end{align*}$$

(3.8a)

(3.8b)

In analogy with the derivation in Section 2.2.2 the solution to (3.8) is on the arc

$$p(\mu) = -[\hat{H}_k + \mu I_p]^{-1}\hat{g}_k.$$  

(3.9)

It remains to estimate $\mu_k$ such that $p(\mu_k)$ solves 3.8. By [26, Lemma 6.4.1], there is a unique $\mu_k \geq 0$ such that $||p(\mu_k)|| = \rho_k$, unless $||p(0)|| \leq \rho_k$ in which case $\mu_k = 0$. Estimating $\mu_k$ is difficult, but well understood. Chapter 7 in Conn, Gould, and Toint [18] is a comprehensive treatment. In particular, Algorithm 7.3.6 in [18] is robust and easily implemented.

Now consider the alternative trust region subproblem:

$$\begin{align*}
\min_{p} \quad & \hat{g}_k(p) = \hat{g}_k^T p + \frac{1}{2} p^T \hat{H}_k p \\
\text{s.t.} \quad & ||p||_{W_k} \leq \rho_k
\end{align*}$$

(3.10a)

(3.10b)

where we use the norm $||p||_{W_k} = (p^T W_k p)^{1/2}$ in the constraint. This corresponds to the problem

$$\begin{align*}
\min_{\theta} \quad & \hat{g}_k^T (\theta - \theta_k) + \frac{1}{2} (\theta - \theta_k)^T \hat{H}_k (\theta - \theta_k) \\
\text{s.t.} \quad & \theta \in E_k(\rho_k),
\end{align*}$$

(3.11a)

(3.11b)
i.e., we minimize the quadratic model subject to an ellipsoidal constraint concentric with the ellipsoidal design region. This problem is addressed in [26, Chapter 7]. The solution is on the arc

$$\theta_k - [\hat{H}_k + \mu W_k]^{-1}\hat{g}_k.$$ (3.12)

The $\mu_k$ that solves the optimization problem can be estimated by the same algorithms as the unscaled case.

A comprehensive convergence theory exists for trust region methods in the deterministic setting [18]. The sufficient conditions for convergence prescribe approaches to update $\rho_k$ that work well in practice. One such scheme is described in [18, Chapter 17]. Fix constants $\eta_1, \eta_2$ satisfying $0 < \eta_1 \leq \eta_2 < 1$ and $c_1, c_2$ satisfying $0 < c_2 < 1 \leq c_1$. Compute

$$r_k = \frac{f(\theta_k) - f(\theta_k + p)}{m(\theta_k) - m(\theta_k + p)}$$

and update the trust region according to:

$$\rho_{k+1} = \begin{cases} 
\max (c_1 \|p\|W_k, \rho_k) & r_k \geq \eta_2 \\
\rho_k & r_k \in [\eta_1, \eta_2) \\
c_2\|p\|W_k & r_k < \eta_1.
\end{cases}$$

The authors suggest $\eta_1 = 0.05$, $\eta_2 = 0.9$, $c_1 = 2.5$, and $c_2 = 0.25$ based on years of experience and numerous numerical experiments. A similar approach could be applied in the stochastic setting. Chang, Hong, and Wan [10] suggested such a scheme for the STRONG algorithm where estimates of the function are replaced by the mean of multiple observations. However, simply choosing more conservative constants is another approach that does not require replicated observations.
3. Implicit Trust Region Radii

In Section 3.3.1, we describe an approach where the trust region radius \( \rho_k \) is fixed and we estimate \( \mu_k \). In this section we describe a different approach. As \( \mu \) increases, the length of the step 
\[-[\hat{H}_k + \mu W_k]^{-1}\hat{g}_k \]
decreases. In analogy with the decreasing step length control sequence described for SA methods in Section 2.1 we can fix a sequence \( \{\mu_k\}_{k=0}^{\infty} \) such that step sizes tend smaller. Indeed, if the standard Euclidean norm is used in the constraint as in (3.8b) and \( \hat{H}_k = [0] \) then the algorithm (3.3) is \( \theta_k = \theta_{k-1} - \mu_k^{-1}\hat{g}_k \), i.e., it is the standard SA algorithm with the gradient estimate \( \hat{g}_k \) and \( a_k = \mu_k^{-1} \) (compare with (2.1)).

In this approach, the trust region radius is implicit because \( \mu_k \) solves the trust region subproblem for some \( \rho_k \), i.e., for any \( \mu_k > 0 \) there exists a unique trust region radius \( \rho_k \) such that

\[
\|-[\hat{H}_k + \mu_k W_k]^{-1}\hat{g}_k\|_W = \rho_k.
\]

Figure 3.1 demonstrates this phenomenon. A trust region is depicted by the ellipse centered at \( \theta_k \) and the contours of the objective of the quadratic model are depicted by the ellipses centered at \( \theta_k - \hat{H}_k^{-1}\hat{g}_k \). As \( \mu_k > 0 \) goes to \( \infty \) the corresponding trust region constrained minimizers form the arc in the figure from \( \theta_k - \hat{H}_k^{-1}\hat{g}_k \) to \( \theta_k \).

One approach to updating \( \mu_k \) is to fix constants \( c \geq 0 \) and \( d > 0 \) and use

\[
\mu_k = d(c + k + 1).
\]

The QNSTOP-BASIC algorithm, as described in Section 3.6, uses this updating scheme. The convergence theory in Chapter 4 motivates this particular choice.
3. QNSTOP

3.4 Updating the Experimental Design Region

In this section we describe the QNSTOP approach to constructing the ellipsoidal design regions used in Section 3.1. Stablein, Carter, and Wampler [79] considered confidence regions for the constrained minimizer of a quadratic model fit by regression. In an early description of the QNSTOP approach, Trosset [86] described a convenient ellipsoidal approximation of the confidence set for the minimizer of a quadratic subject to a trust region constraint.

First, consider a quadratic model fit by linear regression. Let $X^q_k$ denote the $N \times (1+p+p(p+1)/2)$ design matrix for fitting a the quadratic model. Let $\xi_k$ denote the $1+p+p(p+1)/2$ length vector of coefficients and denote the corresponding step $s_k = \theta_{k+1} - \theta_k$ obtained by one of the methods in Section 3.3. Define the matrix $A$ such that $\nabla \hat{m}_k(s_k) = \hat{g}_k + \hat{H}_ks_k = A\xi_k$. The covariance matrix of $\xi_k$ is $\sigma^2((X^q_k)^T(X^q_k))^{-1}$ where $\sigma^2$ is the OLS estimate of the variance; hence, the covariance matrix
of $A\xi_k$ is $\sigma^2 A((X_q^k)^\top(X_q^k))^{-1}A_k^\top$.

Let $\delta(s) = [\hat{H}_k + \mu_k W_k]s + \hat{g}_k$, an estimated stationary point of the Lagrangian, and $V = \sigma^2 A((X_q^k)^\top(X_q^k))^{-1}A_k^\top$. The set of $s$ that satisfy

$$
\delta(s)^\top V^{-1} \delta(s) \leq \chi_{p,1-\alpha}^2
$$

is an approximation of the $1 - \alpha$ percentile confidence set for the minimizer. Then

$$
E_{k+1}(\chi_{p,1-\alpha}) \triangleq \{ x \in \mathbb{R}^p : (x - \theta_k)^\top W_k (x - \theta_k) \leq \chi_{p,1-\alpha}^2 \},
$$

where

$$
W_{k+1} = \left( \hat{H}_k + \mu_k W_k \right)^\top V^{-1} \left( \hat{H}_k + \mu_k W_k \right),
$$

is an ellipsoidal approximation of the confidence set.

However, if a linear model is fit by linear regression and the model Hessian matrix updated by a secant update then the approach differs. For sake of simplicity, assume we obtain the gradient by fitting a linear model by linear regression and use the conventional SR1 secant update to obtain the model Hessian matrix. Let $X_k$ denote the design matrix for computing the gradient of the linear model. Then

$$
\nabla \hat{m}_k(s_k) = \hat{g}_k + \hat{H}_ks_k
= \hat{g}_k + \hat{H}_{k-1}s_k + U_k^s s_k
= \hat{g}_k + \hat{H}_{k-1}s_k + \frac{(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})^\top}{(\nu_{k-1} - \hat{H}_{k-1}s_{k-1})^\top s_{k-1}}s_k.
$$

47
Assume $s_k \approx s_{k-1}$ then

$$
\nabla \hat{m}_k(s_k) \approx \hat{g}_k + \hat{H}_{k-1}s_k + (\nu_{k-1} - \hat{H}_{k-1}s_{k-1})
$$

$$
\approx \hat{g}_k + \nu_{k-1}
$$

$$
= 2\hat{g}_k - \hat{g}_{k-1}.
$$

The assumption that $s_k \approx s_{k-1}$ is not too unreasonable when the respective iterates are far from a solution. The length of the two vectors should be comparable if the trust region radius did not change significantly. Moreover, when the respective iterates are far from the solution it is not unreasonable to assume the steps are nearly in the same direction. Then

$$
V = 4\sigma^2(X_k^T X_k)^{-1}
$$

(3.13)

denotes an approximation for the covariance matrix of $\nabla \hat{m}_k(s_k)$ and again we compute

$$
W_{k+1} = \left( \hat{H}_k + \mu_k W_k \right)^T V^{-1} \left( \hat{H}_k + \mu_k W_k \right)
$$

In the case of the CSR1 update we use the same approximation (3.13) for the covariance matrix.

Strictly using the updates for $W_{k+1}$ above can lead to degenerate ellipsoids. To obtain useful design ellipsoids we enforce the constraints $\gamma^{-1}I_p \preceq W_{k+1} \preceq \gamma I_p$ and $\det(W_{k+1}) = 1$ by modifying the eigenvalues; hence, the definition of $W_\gamma$. The update can be written succinctly as follows:

**Algorithm 3.2.** $W_{k+1}$ Update

1. Set $W_{k+1}^* = \left[ \hat{H}_k + \mu_k W_k \right] (X_k^T X_k) \left[ \hat{H}_k + \mu_k W_k \right]$.

2. Let $\lambda_1^* \leq \ldots \leq \lambda_p^*$ denote the eigenvalues of $W_{k+1}^*$ with respective eigenvectors $v_1, \ldots, v_p$. Set
3. Set \( W_{k+1} = \alpha \sum_{i=1}^{p} \lambda_i v_i v_i^T \) where \( \alpha > 0 \) is chosen such that \( \det(W_{k+1}) = 1 \).

### 3.5 Stopping Criteria

A typical stopping criterion in the stochastic setting is the expenditure of a budget of function evaluations. In this scenario, the user has selected a budget that has been chosen according to wall clock time constraints, financial limitations, physical constraints, or the budget may even be chosen arbitrarily. In the context of the general stochastic optimization problem described in Chapter 1, however, one must be clear in defining the budget. The budget should reflect the total number of samples used to observe all function evaluations instead of simply the total number of function evaluations. The distinction is of key importance because one could carefully allocate samples to later function evaluations if the budget is known in advance. A comprehensive treatment of the tradeoff between samples per function evaluation and function evaluation per iteration would itself be a serious undertaking. To simplify matters in the comments that follow we consider only the scenario where a fixed \( n \) samples are observed for each function evaluation and the budget can fairly reflect the number of function evaluations.

In the context of a fixed budget, there is little an algorithm designer can do short of iterating until the budget is exhausted. However, motivated by traditional response surface methods one option is to stop the algorithm short of exhausting the budget such that a quadratic or even more sophisticated model can be fit using the remaining the function evaluations. For example, one could terminate the iterative algorithm with a sufficient number of function evaluations remaining in the budget to construct a second-order design in the current ellipsoidal region to fit a quadratic model. So long as the dimension of the problem does not preclude the use of such costly designs.
one might conjecture that this is often the best use of the remainder of the budget.

3.6 A Basic QNSTOP Algorithm

In Sections 3.1-3.4 we describe the components of a QNSTOP method. In this section, we describe QNSTOP-BASIC, a basic QNSTOP algorithm using the generalized central differencing designs, the CSR1 quasi-secant update, and implicitly defined trust region radii. We assume that a budget \(B\) of function evaluations is available and terminate when the subsequent iteration would exceed \(B\). We use choices of \(\{\tau_k\}_{k=0}^{\infty}, \{\mu_k\}_{k=0}^{\infty}, \eta, \) and \(\gamma\) that satisfy the sufficient conditions for convergence described in Chapter 4. We fix \(a > 0\) and \(b \in (0, 1/2)\) and use the sequence defined by

\[
\tau_k = a(k + 1)^{-b}
\]

to control the size of the design regions. We fix \(c \geq 0\) and \(d > 0\) and use

\[
\mu_k = d(c + k + 1)
\]

for step length control. Lastly, \(d > 0, \eta \geq 0, \) and \(\gamma \geq 1\) must satisfy

\[
d > \eta \gamma.
\]

Algorithm 3.3. QNSTOP-BASIC

**Step 0 (initialization):** Fix \(a > 0, \ b \in (0, 1/2), \ c \geq 0, \ d > 0, \ \eta \geq 0, \ \gamma \geq 1, \ N = 2p, \) and \(\Pi > 0.\) Choose an initial iterate \(\theta_0,\) scaling matrix \(W_0,\) and model Hessian matrix \(\hat{H}_0\) such that

\[-\eta(c + 1)I_p \leq \hat{H}_0 \leq \eta(c + 1)I_p.\]

Set \(k = 0.\)
3. QNSTOP

Step 1 (regression experiment) : Let \( \tau_k = a(k+1)^{-b} \). Construct the generalized central differencing design \( D_k = \{x_{k1}, \ldots, x_{kN}\} \subset E_k(\tau_k) \). Observe the response vector \( Y_k = (y_{k1}, \ldots, y_{kN})^T \).

Compute \( \tilde{g}_k \) by solving the linear system \( (X_k^T X_k) \hat{g}_k = X_k^T Y_k \).

Step 2 (quasi-secant update) : If \( k > 0 \), compute the model Hessian matrix \( \hat{H}_k \) using CSR1.

Step 3 (update iterate) : Let \( \mu_k = d(c + k + 1) \), solve \( [\hat{H}_k + \mu_k W_k] s_k = -\tilde{g}_k \), and compute \( \theta_{k+1} = \theta_k + s_k \).

Step 4 (update subsequent design ellipsoid) : Compute \( W_{k+1} \in W_\gamma \) using Algorithm 3.2.

Step 5 If \( (k+2)N < B \) then increment \( k \) by 1 and go to Step 1. Otherwise, the algorithm terminates.

In Chapter 4, we discuss convergence theory for quasi-Newton methods for stochastic optimization. QNSTOP-BASIC does not satisfy the requirements as-is. However, by obtaining two independent estimates of the gradient; an additional one for use exclusively for the quasi-secant update, this algorithm will have favorable convergence guarantees. We refer to this variant as QNSTOP-BASIC-2.
4

Convergence Theory

The convergence theory developed in this chapter generalizes results established for stochastic approximation methods to quasi-Newton methods for stochastic optimization. A significant number of papers and books have been written on the convergence theory of stochastic approximation methods. Spall [78, Section 4.3.1] described two prototypical sets of sufficient conditions for establishing almost sure convergence: the “statistics” conditions and the “engineering” conditions. These types of conditions apply to both Robbins-Monro type methods for root-finding and Kiefer-Wolfowitz type methods for optimization. In this chapter we consider only the optimization problem.

The statistics approach considers the behavior of the algorithm as a result of observing noisy function evaluations to estimate the gradient. The engineering approach defines an ordinary differential equation that asymptotically behaves similarly to the stochastic approximation method. Fabian [29] established almost sure convergence for the Kiefer-Wolfowitz algorithm using statistics type sufficient conditions. In this chapter, we develop an almost sure convergence theory for quasi-Newton methods by generalizing Fabian’s results. Section 4.1 describes terminology and notation used throughout the chapter. Section 4.2 duplicates a key lemma in [29] and proves a corollary of...
4. Convergence Theory

Fabian’s convergence theorem that generalizes it to methods using regression models for estimating the gradient. Section 4.3 uses the aforementioned corollary to establish sufficient conditions for convergence of a general class of quasi-Newton methods for stochastic optimization. This theory is demonstrated by applying it to QNSTOP-BASIC-2.

4.1 Background and Notation

Let $(\Omega, \mathcal{F}, P)$ denote a probability space, i.e., $\Omega$ is a sample space, $\mathcal{F}$ is a $\sigma$-algebra of subsets of $\Omega$, and $P$ is a probability measure on $(\Omega, \mathcal{F})$. All random variables are defined on $(\Omega, \mathcal{F}, P)$. We study the minimization of the function $f : \mathbb{R}^p \mapsto \mathbb{R}$ with unique minimum $\theta^*$. We observe the noisy evaluations $Y_x = f(x) + \epsilon_x$ where $\epsilon_x$ is a random variable representing the noise. Let $x_{ki}$ denote the $i$th design site in iteration $k$; hence, we observe

$$Y_{x_{ki}} = f(x_{ki}) + \epsilon_{x_{ki}}.$$ 

Let $\theta_k$ and $g_k$ denote random $p$-dimensional vectors. The sequence $\{\theta_k\}_{k=0}^\infty$ denotes the iterates of the algorithm and the sequence $\{g_k\}_{k=0}^\infty$ denotes estimates of the gradient of $f$ at the corresponding iterates. We observe $\{Y_{x_{ki}}\}_{i=1}^{N_k}$ at design sites $\{x_{k1}, \ldots, x_{kN_k}\}$ in a neighborhood of $\theta_k$. The design sites can either be random or pre-determined given $\theta_k$. The random vector $g_k$ is a function of both the design sites and the observations. Let $m_k$ and $M_k$ be positive numbers. We consider algorithms of the form

$$\theta_{k+1} = \theta_k - B_k g_k$$  \hspace{1cm} (4.1)$$

where $B_k$ is a symmetric and positive definite matrix-valued measurable function with eigenvalues surely in the interval $[m_k, M_k]$. Let $\mathbb{E}_k[\cdot]$ denote expectation conditioned on $\mathcal{F}_k \subset \mathcal{F}$, the $\sigma$-algebra
4. Convergence Theory

generated by \{\theta_0, g_0, \ldots, g_{k-1}, B_0, \ldots, B_k\}.

We decompose the errors in the estimates of the gradient. Following the terminology in [53] we denote the “systematic error”, i.e., error due to the method used to estimate the gradient, by

\[ \beta_k = \nabla f(\theta_k) - E_k[g_k]. \] (4.2)

We denote the “stochastic error”, i.e., error due to chance variation in the function evaluations,

\[ \delta_k = E_k[g_k] - g_k. \] (4.3)

Noting that

\[ \beta_k + \delta_k = \nabla f(\theta_k) - g_k \]

we write (4.1) as

\[ \theta_{k+1} = \theta_k - B_k \nabla f(\theta_k) + B_k \beta_k + B_k \delta_k. \] (4.4)

Fabian [29] considers the case using the central differencing estimate of the gradient, i.e.,

\[ g_k = \frac{1}{2c_k} \begin{bmatrix} Y_{\theta_k+c_k e_1} - Y_{\theta_k-c_k e_1} \\ \vdots \\ Y_{\theta_k+c_k e_p} - Y_{\theta_k-c_k e_p} \end{bmatrix} \]

where \( e_i \) is the unit vector in the \( i \)th coordinate direction. Moreover, Fabian considers the algorithm \( \theta_{k+1} = \theta_k - \alpha_k B_k g_k \) for some sequence \{\alpha_k\} of positive numbers. We present the case where \( \alpha_k = 1 \) for all \( k \) and we consider more general gradient estimates. Unless noted otherwise \( \| \cdot \| \) denotes the Euclidean norm for both vectors and matrices. The operators \( \lambda_{\min} (\lambda_{\max}) \) and \( \sigma_{\min} (\sigma_{\max}) \) denote the minimum (maximum) eigenvalues and singular values, respectively. Let \( I_p \) denote the \( p \times p \)
identity matrix.

4. Convergence Theory

4.2 An Almost Sure Convergence Theorem

Fabian [29] proved the following convergence lemma for iterative algorithms.

Lemma 4.1. ([29, Lemma 3.3]) Suppose $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is bounded from below, the Hessian exists and $\|\nabla^2 f(\theta)\| \leq L$ for some $L > 0$ and all $\theta \in \mathbb{R}^p$. Let $\theta_k$ and $u_k$ denote $p$-dimensional random vectors, $\mathcal{G}_k$ be the $\sigma$-algebra generated by $\{\theta_0, \ldots, \theta_k\}$,

$$\theta_{k+1} = \theta_k - u_k$$

and $a_k, b_k, c_k, d_k$ nonnegative numbers, $a_k > 0$ and $D_k$ nonnegative random variables. Let $z_k = \mathbb{E}[u_k|\mathcal{G}_k]$.

If

$$z_k^T \nabla f(\theta_k) \geq a_k D_k^2 - b_k - c_k D_k$$

$$\mathbb{E}[\|u_k\|^2|\mathcal{G}_k] \leq b_k + c_k D_k + d_k D_k^2$$

and

$$\sum a_k = \infty, \sum b_k < \infty, \sum \frac{c_k^2}{a_k} < \infty, \frac{c_k}{a_k} \rightarrow 0, \frac{d_k}{a_k} \rightarrow 0,$$

then a subsequence of $\{D_k\}$ converges to 0 and the sequence $\{f(\theta_k)\}$ converges to a random variable.

[29, Theorem 2.4] establishes the almost sure convergence of the iterates of a generalization of the Kiefer-Wolfowitz algorithm. The proof relies on Lemma 4.1. In this section we prove a corollary of [29, Theorem 2.4] extending the result to more general gradient estimators. The proof is nearly identical to that in [29, Theorem 2.4], however, we fix an insignificant error and modify it to include
more general gradient estimates.

**Corollary 4.2.** Suppose \( f : \mathbb{R}^p \to \mathbb{R} \) is bounded from below, the Hessian exists and \( \| \nabla^2 f(\theta) \| \leq L \) for some \( L > 0 \) and all \( \theta \in \mathbb{R}^p \). Assume the function \( f \) has a unique minimizer \( \theta^* \) and

\[
\inf_{\| \theta - \theta^* \| \geq \phi} \{ \| \nabla f(\theta) \| \} > 0 \quad \text{and} \quad \inf_{\| \theta - \theta^* \| \geq \phi} \{ \| f(\theta) - f(\theta^*) \| \} > 0
\]

(4.5)

for some \( \phi > 0 \). Let \( \{ \tau_k \}_{k=0}^{\infty} \) denote a sequence of positive numbers. Suppose there are constants \( c_\delta > 0 \) and \( c_\beta > 0 \) such that for all \( k \geq 0 \), \( \mathbb{E}_k[\| \delta_k \|^2] \leq c_\delta \tau_k^{-2} \) and \( \| \beta_k \| \leq c_\beta \tau_k \) and there are sequences of positive numbers \( \{ m_k \}_{k=0}^{\infty} \) and \( \{ M_k \}_{k=0}^{\infty} \) such that

\[
\sum_{k=0}^{\infty} m_k = \infty, \quad \sum_{k=0}^{\infty} M_k \tau_k < \infty, \quad \sum_{k=0}^{\infty} M_k^2 \tau_k^{-2} < \infty,
\]

(4.6)

then algorithm (4.1) generates iterates such that \( \theta_k \to \theta^* \) almost surely.

**Proof.** By assumption and (4.2) we have both

\[
\nabla f(\theta_k)^T B_k \mathbb{E}_k [g_k] = \nabla f(\theta_k)^T B_k \nabla f(\theta_k) - \nabla f(\theta_k)^T B_k \beta_k \geq \nabla f(\theta_k)^T B_k \nabla f(\theta_k) - c_\beta \tau_k \| \nabla f(\theta_k) \| \| B_k \| \geq m_k \| \nabla f(\theta_k) \|^2 - c_\beta \tau_k M_k \| \nabla f(\theta_k) \|
\]

(4.7)

and

\[
\| B_k \mathbb{E}_k [g_k] \|^2 = \| B_k \nabla f(\theta_k) - B_k \beta_k \|^2 \leq 2 \| B_k \nabla f(\theta_k) \|^2 + 2 \| B_k \beta_k \|^2 \leq 2 M_k^2 \| \nabla f(\theta_k) \|^2 + 2 c_\beta^2 \tau_k^2 M_k^2.
\]

(4.8)
4. Convergence Theory

By assumption and (4.3) we have

\[
\mathbb{E}_k \left[ \|B_k (g_k - g_k) \|^2 \right] = \mathbb{E}_k \left[ \|B_k \delta_k \|^2 \right] \\
\leq M_k^2 \mathbb{E}_k \left[ \|\delta_k \|^2 \right] \\
\leq c_\delta M_k^2 \tau_k^{-2}.
\] (4.9)

Let \( u_k = B_k g_k, D_k = \|\nabla f(\theta_k)\|, \) and \( z_k = \mathbb{E}_k [u_k] \) then we rewrite (4.7)

\[
z_k^T \nabla f(\theta_k) \geq m_k D_k^2 - c_\beta \tau_k M_k D_k,
\] (4.10)

and combining (4.8) and (4.9) we obtain

\[
\mathbb{E}_k \left[ \|u_k\|^2 \right] = \mathbb{E}_k \left[ \|B_k g_k\|^2 \right] \\
= \mathbb{E}_k \left[ \|B_k g_k - B_k \mathbb{E}_k [g_k] + B_k \mathbb{E}_k [g_k]\|^2 \right] \\
\leq \mathbb{E}_k \left[ 2 \|B_k g_k - B_k \mathbb{E}_k [g_k]\|^2 + 2 \|B_k \mathbb{E}_k [g_k]\|^2 \right] \\
\leq 2c_\delta M_k^2 \tau_k^{-2} + 4M_k^2 D_k^2 + 4c_\beta \tau_k^2 M_k^2.
\] (4.11)

Let \( a_k = m_k, b_k = 2c_\delta M_k^2 \tau_k^{-2} + 4c_\beta \tau_k^2 M_k^2, c_k = c_\beta \tau_k M_k, \) and \( d_k = 4M_k^2 \) then we rewrite (4.10) and (4.11) as

\[
z_k^T \nabla f(\theta_k) \geq a_k D_k^2 - c_k D_k \\
\geq a_k D_k^2 - b_k - c_k D_k
\]
and

\[ E_k[\|u_k\|^2] \leq b_k + d_k D_k^2 \]
\[ \leq b_k + c_k D_k + d_k D_k^2. \]

We then apply Lemma 4.1; hence, a subsequence of \{\|\nabla f(\theta_k)\|\} converges to 0 almost surely and \( f(\theta_k) \) converges almost surely. Then, by the assumption (4.5), \( \theta_k \to \theta^* \) almost surely as \( k \to \infty \).

4.3 Almost Sure Convergence Theory for QNSTOP

This section establishes almost sure convergence theory for a class of quasi-Newton methods for stochastic optimization. Let

\[ \mathcal{W}_\gamma \triangleq \{ W \in \mathbb{R}^{p \times p} : W = W^T, \det(W) = 1, \gamma^{-1} I_p \preceq W \preceq \gamma I_p \} \]

for some \( \gamma \geq 1 \). Let \( H_k \) and \( W_k \) denote \( p \times p \) symmetric matrix-valued measurable functions. We consider algorithms of the form

\[ \theta_{k+1} = \theta_k - [H_k + \mu_k W_k]^{-1} g_k \] (4.12)

where \( \mu_k > 0 \). In particular, we study algorithms where

\[ -\eta I_p \preceq H_k - H_{k-1} \preceq \eta I_p \] for \( k = 1, 2, \ldots \) (4.13)

for some \( \eta \geq 0 \) and

\[ W_k \in \mathcal{W}_\gamma \] for \( k = 0, 1, \ldots \) (4.14)
are both satisfied surely, i.e., for each $\omega \in \Omega$.

Theorem 4.3 prescribes sufficient conditions such that algorithms of the form (4.12) have favorable convergence guarantees.

**Theorem 4.3.** Suppose we fix $\gamma \geq 1$, $\eta \geq 0$, and $c \geq 0$ and choose $H_0$ such that

$$-\eta(c+1) \preceq H_0 \preceq \eta(c+1)$$

holds surely. If (4.13) and (4.14) are satisfied surely, then there exist positive numbers $\mu_k$, $\tau_k$, $m_k$, and $M_k$ such that for $k = 0, 1, \ldots$

$$m_k I_p \preceq [H_k + \mu_k W_k]^{-1} \preceq M_k I_p$$

holds surely,

$$\sum_{k=0}^{\infty} m_k = \infty, \quad \sum_{k=0}^{\infty} M_k \tau_k < \infty, \quad \sum_{k=0}^{\infty} M_k^2 \tau_k^{-2} < \infty,$$  \hspace{1cm} (4.15)

and

$$\tau_k \to 0, (\tau_k + M_k) M_k m_k^{-1} \to 0,$$  \hspace{1cm} (4.16)

as $k \to \infty$.

**Proof.** Fix $a > 0$, $b \in (0, 1/2)$, and $d > 0$ such that $d > \eta \gamma$. We choose the sequences

$$\mu_k = d(c+k+1)$$

$$\tau_k = a(k+1)^{-b}$$

$$m_k = [\mu_k(\eta d^{-1} + \gamma)]^{-1}$$

$$M_k = [\mu_k(-\eta d^{-1} + \gamma^{-1})]^{-1}$$
which immediately satisfy (4.15) and (4.16).

Each statement for the remainder of the proof involving \( H_k \) and/or \( W_k \) holds surely. By construction, \( \lambda_{\min} \left( \frac{1}{c+1} H_0 \right) \geq -\eta \) and \( \lambda_{\max} \left( \frac{1}{c+1} H_0 \right) \leq \eta \). Define \( U_k = H_k - H_{k-1} \). We first observe that \( \lambda_{\min}(H_k) \geq \lambda_{\min}(H_{k-1}) + \lambda_{\min}(U_k) \). Then for \( k = 1, 2, \ldots \)

\[
\lambda_{\min} \left( \frac{d}{\mu_k} H_k \right) \geq \lambda_{\min} \left( \frac{d}{\mu_k} H_{k-1} \right) + \lambda_{\min} \left( \frac{d}{\mu_k} U_k \right) = \frac{c+k}{c+k+1} \lambda_{\min} \left( \frac{1}{c+k} H_{k-1} \right) + \frac{1}{c+k+1} \lambda_{\min} \left( U_k \right) \geq \frac{c+k}{c+k+1} (-\eta) + \frac{1}{c+k+1} (-\eta) \geq -\eta
\]

and similarly for \( k = 1, 2, \ldots \)

\[
\lambda_{\max} \left( \frac{d}{\mu_k} H_k \right) \leq \lambda_{\max} \left( \frac{d}{\mu_k} H_{k-1} \right) + \lambda_{\max} \left( \frac{d}{\mu_k} U_k \right) = \frac{c+k}{c+k+1} \lambda_{\max} \left( \frac{1}{c+k} H_{k-1} \right) + \frac{1}{c+k+1} \lambda_{\max} \left( U_k \right) \leq \frac{c+k}{c+k+1} (\eta) + \frac{1}{c+k+1} (\eta) \leq \eta.
\]

Recall that \( \gamma^{-1} I_p \preceq W_k \preceq \gamma I_p \). So,

\[
\frac{d}{\mu_k} \lambda_{\max}(H_k + \mu_k W_k) \leq \eta + d \gamma
\]

and

\[
\frac{d}{\mu_k} \lambda_{\min}(H_k + \mu_k W_k) \geq -\eta + d \gamma^{-1}.
\]

Finally, \( d > \eta \gamma \) implies that \( -\eta + d \gamma^{-1} > 0 \) so \( (H_k + \mu_k W_k) \) is symmetric and positive definite.
4. Convergence Theory

Therefore,

\[
[H_k + \mu_k W_k]^{-1} \preceq \frac{1}{\mu_k (-\eta d^{-1} + \gamma^{-1})} I_p = M_k I_p
\]

and

\[
[H_k + \mu_k W_k]^{-1} \succeq \frac{1}{\mu_k (\eta d^{-1} + \gamma)} I_p = m_k I_p.
\]

\[\square\]

Lemmas 4.4 and 4.5 below establish the remaining conditions in Corollary 4.2 for methods that use \Pi\)-poised designs and estimate the gradient by fitting a linear model by least squares. Specifically, we establish sufficient conditions such that there are constants \(c_\delta > 0\) and \(c_\beta > 0\) such that for \(k \geq 0\), \(\mathbb{E}_k[\|\delta_k\|^2] \leq c_\delta \tau_k^{-2}\) and \(\|\beta_k\| \leq c_\beta \tau_k\). Lemma 4.4 follows a similar technique to that used in Theorem 2.13 in [20, Errata]. However, before continuing we explicitly state the “statistics” assumptions for our context.

### Assumptions

1. \(f\) is twice continuously differentiable, bounded from below, and \(\|\nabla^2 f(x)\| \leq L\) for some \(L > 0\) and all \(x \in \mathbb{R}^p\).

2. The errors in the function evaluations have mean zero and finite variance, i.e., \(\mathbb{E}_k[\epsilon_k] = 0\), and \(\mathbb{E}_k[\epsilon_k^2] \leq \epsilon_c\) for some constant \(\epsilon_c > 0\) where \(\mathbb{E}_k[\cdot]\) is an appropriate expectation.

3. The function \(f\) has a unique minimizer \(\theta^*\) and \(\inf_{\|x-\theta^*\| > \phi} \{\|\nabla f(x)\|\} > 0\) and \(\inf_{\|x-\theta^*\| > \phi} \{|f(x) - f(\theta^*)|\} > 0\) for some \(\phi > 0\).
Let \( D_k = \{x_{k1}, \ldots, x_{kN}\} \) denote the experimental design in iteration \( k \). Typically, the design is pre-determined given \( \theta_k \), however, it is possible to accommodate random designs. To do so, we extend \( F_k \) to be the sigma algebra generated by \( \{\theta_0, g_0, \ldots, g_{k-1}, B_0, \ldots, B_k, D_0, \ldots, D_k\} \) and ensure that inequality (4.17) below holds surely. We continue the analysis as if \( D_k \) is predetermined given \( \theta_k \). Recall the notation from Chapter 3. Let 

\[
\bar{x}_k = \sum_{i=1}^N x_{ki},
\]

\[
X_k \triangleq X(D_k) = \begin{bmatrix}
(x_{k1} - \bar{x}_k)^T \\
\vdots \\
(x_{kN} - \bar{x}_k)^T
\end{bmatrix},
\]

\[
\Delta_k = 2\tau_k \gamma^{1/2}, \text{ and } \hat{X}_k = \frac{1}{\Delta_k} X_k. \]

If \( \sigma_{\min}(\hat{X}_k) \geq \Pi \) and \( \sigma_{\max}(\hat{X}_k) \leq 1 \) we say \( D_k = \{x_{k1}, \ldots, x_{kN}\} \) is \( \Pi \)-poised. Let \( Y_k = (Y_{x_{k1}}, \ldots, Y_{x_{kN}})^T \). We denote the Moore-Penrose generalized inverse by superscript \( \dagger \), which, for a matrix \( A \) with full column rank, we can write \( A^\dagger = (A^T A)^{-1} A^T \). Then,

\[
\|X_k^\dagger\| = \| (X_k^T X_k)^{-1} X_k \| \\
\leq \| (X_k^T X_k)^{-1} \| \| X_k \| \\
= \| \Delta_k^{-2} (\hat{X}_k^T \hat{X}_k)^{-1} \| \| \Delta_k \hat{X}_k \| \\
= \frac{1}{\Delta_k} \| (\hat{X}_k^T \hat{X}_k)^{-1} \| \| \hat{X}_k \| \\
\leq \frac{1}{\Delta_k \Pi^2} \tag{4.17}
\]

**Lemma 4.4.** If Assumptions 1 and 2 hold then there exists \( c_\beta > 0 \) such that for \( k \geq 0 \), \( \| \beta_k \| \leq c_\beta \tau_k \).

**Proof.** Let \( r_i = f(x_k) - f(x_{ki}) + (x_{ki} - \bar{x}_k)^T \nabla f(x_k) \). By the boundedness of the Hessian we have 

\[
|r_i| \leq \frac{L}{2} \| x_{ki} - \bar{x}_k \|^2 \leq \frac{L}{2} \Delta_k^2. \]

Let \( r = (r_1, \ldots, r_N)^T \) and \( 1_N \) denote the \( N \) vector \( (1, \ldots, 1)^T \). We can
write the system
\[ \mathbf{r} = [\mathbf{1}_N \quad \mathbf{X}_k] \begin{bmatrix} f(\overline{x}_k) \\ \nabla f(\overline{x}_k) \end{bmatrix} - \mathbb{E}_k [\mathbf{Y}_k]. \]

Left multiply both sides by \([\mathbf{1}_N \quad \mathbf{X}_k]^\top\):
\[ [\mathbf{1}_N \quad \mathbf{X}_k]^\top \mathbf{r} = \begin{bmatrix} f(\overline{x}_k) \\ \nabla f(\overline{x}_k) \end{bmatrix} - [\mathbf{1}_N \quad \mathbf{X}_k]^\top \mathbb{E}_k [\mathbf{Y}_k]. \]

Let \(\mathbf{J}_N = \mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^\top\). Because \(\mathbf{X}_k\) is column centered, \(\mathbf{X}_k = \mathbf{J} \mathbf{X}_k\). So,
\[ [\mathbf{1}_N \quad \mathbf{J} \mathbf{X}_k]^\top [\mathbf{1}_N \quad \mathbf{J} \mathbf{X}_k] = \begin{bmatrix} N & \mathbf{0}_N^\top \\ \mathbf{0}_N & \mathbf{X}_k^\top \mathbf{J}^\top \mathbf{J} \mathbf{X}_k \end{bmatrix} = \begin{bmatrix} N & \mathbf{0}_N^\top \\ \mathbf{0}_N & \mathbf{X}_k^\top \mathbf{X}_k \end{bmatrix}; \]

hence, the inverse is
\[ \left([\mathbf{1}_N \quad \mathbf{J} \mathbf{X}_k]^\top [\mathbf{1}_N \quad \mathbf{J} \mathbf{X}_k]\right)^{-1} = \begin{bmatrix} N^{-1} & \mathbf{0}_N^\top \\ \mathbf{0}_N & (\mathbf{X}_k^\top \mathbf{X}_k)^{-1} \end{bmatrix}. \]

Using this fact, we eliminate the first entry in the vectors on each side of the equality so
\[ \mathbf{X}_k^\top \mathbf{r} = \nabla f(\overline{x}_k) - \mathbf{X}_k^\top \mathbb{E}_k [\mathbf{Y}_k]. \]
Because $g_k$ is the least squares solution $E_k[g_k] = X_k^\dagger E_k[Y_k]$. Then we obtain

$$
\| \nabla f(\mathbf{x}) - E_k[g_k] \| = \left\| \nabla f(\mathbf{x}) - X_k^\dagger \epsilon_k \right\| \\
\leq ||X_k^\dagger|| \| \epsilon_k \| \\
\leq \left( \frac{1}{\Delta_k \Pi^2} \right) \left( \frac{NL\Delta_k^2}{2} \right) \\
= \left( \frac{NL}{2\Pi^2} \right) \Delta_k.
$$

Lastly,

$$
\| \nabla f(\theta_k) - E_k[g_k] \| \leq \| \nabla f(\theta_k) - \nabla f(\mathbf{x}) \| + \| \nabla f(\mathbf{x}) - E_k[g_k] \| \\
\leq L\Delta_k + \left( \frac{NL}{2\Pi^2} \right) \Delta_k \\
= \left( L + \frac{NL}{2\Pi^2} \right) \Delta_k
$$

so $\| \delta_k \| = \| \nabla f(\theta_k) - E_k[g_k] \| \leq \left( L + \frac{NL}{2\Pi^2} \right) \Delta_k = \left( L + \frac{NL}{2\Pi^2} \right) 2\gamma^{1/2} \tau_k$. 

**Lemma 4.5.** If Assumptions 1 and 2 hold then there exists $c_\delta > 0$ such that for $k \geq 0$, $E_k \left[ \| \delta_k \|^2 \right] \leq c_\delta \tau_k^{-2}$.

**Proof.** Let $\epsilon_k = E_k[Y_k] - Y_k$. Then,

$$
E_k \left[ \| E_k[g_k] - g_k \|^2 \right] = E_k \left[ \left\| X_k^\dagger E_k[Y_k] - X_k^\dagger Y_k \right\|^2 \right] \\
= E_k \left[ \left\| X_k^\dagger \epsilon_k \right\|^2 \right] \\
\leq \left\| X_k^\dagger \right\|^2 E_k \left[ \| \epsilon_k \|^2 \right] \\
\leq \left( \frac{1}{\Delta_k \Pi^2} \right)^2 Nc_e \\
= \left( \frac{Nc_e}{4\Pi^2} \right) \tau_k^{-2}.
$$
4. Convergence Theory

To apply the aforementioned theory to QNSTOP-BASIC, it must be modified such that \( B_k = [H_k + \mu_k W_k]^{-1} \) is known, or observed, before \( g_k \) is observed in the sense that \( \mathbb{E}_k[B_k g_k] = B_k \mathbb{E}_k[g_k] \). Because \( H_k \) is a function of the gradient estimate we must observe two independent observations. We observe each design site twice and from one set of observations compute \( g_k^{(1)} \) for computing \( B_k \) and from the other set compute \( g_k^{(2)} \) to estimate \( g_k \) used by QNSTOP-BASIC. We refer to this variant as QNSTOP-BASIC-2.

**Proposition 4.6.** If Assumptions 1, 2, and 3 hold then QNSTOP-BASIC-2 generates iterates such that \( \theta_k \to \theta^* \) almost surely as \( k \to \infty \).

**Proof.** By construction, QNSTOP-BASIC-2 verifies the conditions of Theorem 4.3 and Lemmas 4.4 and 4.5 which in turn verify the conditions for Corollary 4.2. The result follows. \( \square \)
5

Numerical Experiments

In this chapter, we present a variety of experiments demonstrating different aspects of QNSTOP performance. Section 5.1 discusses specific issues that arise in comparing stochastic optimization algorithms. Section 5.2 demonstrates the tradeoffs between interpolation and regression models. Section 5.3 gives numerical results minimizing quasi-convex functions that are observed pointwise with additive Gaussian noise. Section 5.4 gives numerical results for the ambulance base location problem described in Example 1.1. Lastly, Section 5.5 gives numerical results for the tumor model problem described in Example 1.2.

5.1 Comparing Stochastic Optimization Algorithms

Parameters controlling scaling and termination criterion can skew comparisons between any pair of optimization algorithms. Here we demonstrate one such scaling difficulty for comparing the Kiefer-Wolfowitz algorithm and a basic QNSTOP algorithm. Recall that an iteration of the Kiefer-Wolfowitz algorithm is

$$\theta_{k+1} = \theta_k - a_k \overline{g}_k$$
where $a_k$ is a positive number and $g_k$ is a central differencing estimate of the gradient. Consider an iteration of a QNSTOP method:

$$\theta_{k+1} = \theta_k - (H_k + \mu_k W_k)^{-1} g_k$$

where $H_k$ is a model Hessian matrix, $\mu_k$ is a positive number, $W_k$ is a parameter controlling the shape of the ellipsoidal trust region, and $g_k$ is an estimate of the gradient. Moreover, consider a QNSTOP method that uses the central differencing estimate for the gradient, i.e., $g_k = \bar{g}_k$, and has access to the exact Hessian, i.e., $H_k = \nabla^2 f(\theta_k)$. Moreover, the ellipsoidal design region is spherical, i.e., $W_k = I_p$. We write an iteration of this particular QNSTOP algorithm as

$$\theta_{k+1} = \theta_k - \left[\nabla^2 f(\theta_k) + \mu_k I_p\right]^{-1} \bar{g}_k.$$

Consider the optimization of a quadratic function with spherical contours. For all intents and purposes, this is the easiest problem for a QNSTOP algorithm with perfect Hessian information. Without knowing anything about the scale of the problem one might select canonical choices for the gain sequences. Specifically, let $a_k = (1 + k)^{-1}$ and $\mu_k = 1 + k$. Now we rewrite the iteration of the Kiefer-Wolfowitz algorithm as

$$\theta_{k+1} = \theta_k - (1 + k)^{-1} \bar{g}_k$$

and an iteration of the QNSTOP algorithm as

$$\theta_{k+1} = \theta_k - \left[I_p + a_k^{-1} I_p\right]^{-1} g_k$$

$$= \theta_k - (2 + k)^{-1} \bar{g}_k.$$
Given any starting point $\theta_k$, this QNSTOP algorithm takes shorter steps than the Kiefer-Wolfowitz algorithm in the same direction. Hence, if the scaling of the problem is such that the step lengths are short relative to the distance to the minimizer then the QNSTOP method is doomed to make slower progress. This example oversimplifies matters, however, it demonstrates that one must be careful comparing two algorithms without giving special attention to the parameterization.

5.2 Interpolation or Regression?

This dissertation has focused on quasi-Newton methods that estimate the gradient using linear models fit by least squares and subsequently use a quasi-secant or secant update to compute a model Hessian matrix. However, in the early development of the QNSTOP approach we primarily focused on fitting a full quadratic model by least squares. Model-based algorithms for deterministic optimization utilizing a sequence of quadratic models have largely used interpolation to obtain the quadratic model [66, 20]. However, in the context of stochastic optimization it is unclear whether interpolation or regression will typically perform better. Interpolation requires less function evaluations per iteration, however, the variance reduction implicit in regression leads to better quadratic models.

Section 2.3 briefly discussed the Lawera-Thompson, Noisy UOBYQA, and STRONG/STRONG-X algorithms for stochastic optimization. The Noisy UOBYQA algorithm uses interpolation to fit a quadratic model and the Lawera-Thompson and STRONG/STRONG-X algorithms use regression to fit a quadratic model. All three methods use the sample mean of replicated observations at each design site as a pointwise estimator. We compare interpolation models and regression models given $2m$ observations where $m = 1 + p + \binom{p}{2}$, i.e., the number of observations needed to uniquely interpolate a quadratic model.
5. Numerical Experiments

Consider the function \( \phi : \mathbb{R}^p \rightarrow \mathbb{R} \) defined by

\[
\phi(x) = \frac{1}{2} (x - x_0)^T H (x - x_0)
\]

where \( H \) is a symmetric and positive definite matrix and \( x_0 \) is the minimizer. Suppose we observe \( Y_x = \phi(x) + e \) where \( e \sim \text{Normal}(0, s^2) \). Let \( t_0 \) denote an initial guess of the minimizer. We choose a set of \( 2m \) design sites \( \{x_1, \ldots, x_{2m}\} \) uniformly sampled from the ball centered at \( t_0 \) with radius \( r \). Let \( z_i = x_i - t_0 \) and

\[
X_m = \begin{bmatrix}
    z_1^T \\
    z_2^T \\
    \vdots \\
    z_m^T
\end{bmatrix}
\]

denote the matrix where each row is one of the first \( m \) design sites. Let \( \Phi(X_m) \) denote the design matrix for fitting a quadratic model. For example, for \( p = 2 \) the \( i \)th row of \( \Phi(X_m) \) is

\[
\begin{bmatrix}
1 & z_{i1} & z_{i2} & z_{i1}^2 & z_{i1} z_{i2} & z_{i2}^2
\end{bmatrix}
\]

where \( z_i = (z_{i1}, z_{i2})^T \). We sample a design and check to ensure that the condition number of \( \Phi(X_m) \) is less than or equal to 200. Otherwise, we resample until it is. This heuristic precludes designs insufficient for interpolation. We consider three different methods for fitting a quadratic model using \( 2m \) observations:

1. Interpolation (I) – We observe \( Y_x \) twice for each \( i = 1, \ldots, m \). We compute the mean of the two observations at each design site and denote the vector of means \( Y = (Y_1, Y_2, \ldots, Y_m)^T \).

   The interpolated quadratic model defined by \( f_I \), the Hessian matrix \( H_I \), and the gradient \( g_I \)
5. Numerical Experiments

satisfies

\[ Y_i = f_1 + g_i^T (x_i - t_0) + \frac{1}{2} (x_i - t_0)^T H_1 (x_i - t_0). \]

for \( i = 1, \ldots, m \). The minimizer of the quadratic model is

\[ t_1 = t_0 - H_1^{-1} g_i. \]

We record the function evaluation at the minimizer, i.e., \( \phi(t_1) \).

2. Regression with 1 replacement (R1) – We observe \( Y_{x_i} \) twice for each \( i = 1, \ldots, m - 1 \), once for \( i = m \), and once for \( i = m + 1 \). We fit a quadratic model defined by \( f_r, H_r, \) and \( g_r \) to the responses by least squares. As before, the minimizer of the quadratic is

\[ t_r = t_0 - H_r^{-1} g_r \]

and we record the function evaluation at the minimizer, i.e., \( \phi(t_r) \).

3. Regression with \( 2m \) design sites (Rm) – We observe \( Y_{x_i} \) once for \( i = 1, \ldots, 2m \). We fit a quadratic model defined by \( f_R, H_R, \) and \( g_R \) to the responses by least squares. Again, we compute

\[ t_R = t_0 - H_R^{-1} g_R \]

and evaluate \( \phi(t_R) \).

Each run consists of evaluating the function at all three minimizers for a design \( \{x_1, \ldots, x_{2m}\} \). We fix \( H \) and \( s \) and compute 1000 runs. Because the errors are homoskedastic we use the same evaluation errors for each model. If the responses are such that the model Hessian matrix is poorly conditioned then the iterate generated is likely bad. In practice, we prevent taking such steps by
enforcing trust region constraints. In this experiment, we discard the 5% of the function values with greatest magnitude observed by each method. We performed 6 distinct experiments all of which are in \( \mathbb{R}^4 \). In each experiment \( t_0 = (1, 2, 3, 4)^T \) and the minimizer of \( \phi \) is \( x_0 = (-2, -1, 4, 2)^T \). The two choices for \( H \) are

\[
H_1 = \begin{bmatrix}
10 & 0 & 0 & 0 \\
0 & 10 & 0 & 0 \\
0 & 0 & 10 & 0 \\
0 & 0 & 0 & 10
\end{bmatrix} \quad \text{and} \quad H_2 = \begin{bmatrix}
10 & 2 & 3 & -2 \\
2 & 22 & 2 & 10 \\
3 & 2 & 12 & 2 \\
-2 & 10 & 2 & 7
\end{bmatrix}
\]

and choices for \( s \) are \( s_1 = 0.1, s_2 = 1, \) and \( s_3 = 2 \). Tables 5.1 to 5.6 report the five-number summaries (minimum, 1Q, median, 3Q, maximum) for the function values after removing the outliers as described.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0.00063</th>
<th>0.02644</th>
<th>0.06140</th>
<th>0.13379</th>
<th>0.51757</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I )</td>
<td>R1</td>
<td>0.00007</td>
<td>0.01972</td>
<td>0.04146</td>
<td>0.08930</td>
<td>0.33553</td>
</tr>
<tr>
<td>( Rm )</td>
<td>0.00013</td>
<td>0.00549</td>
<td>0.01105</td>
<td>0.01956</td>
<td>0.05183</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Experiment 1 \((H_1, s_1)\) Five-Number Summaries

In each experiment, the regression models are strictly better than the interpolation model with respect to the five-number summary statistics. These experiments address the exploration-exploitation tradeoff. In the interpolation model, exploration is minimal and exploitation is evenly

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0.00086</th>
<th>0.04697</th>
<th>0.13694</th>
<th>0.35243</th>
<th>1.78221</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I )</td>
<td>R1</td>
<td>0.00016</td>
<td>0.03436</td>
<td>0.08913</td>
<td>0.24915</td>
<td>1.43871</td>
</tr>
<tr>
<td>( Rm )</td>
<td>0.00025</td>
<td>0.01035</td>
<td>0.02254</td>
<td>0.04941</td>
<td>0.16355</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Experiment 2 \((H_2, s_1)\) Five-Number Summaries
5. Numerical Experiments

<table>
<thead>
<tr>
<th></th>
<th>Experiment 3 ($H_1$, $s_2$) Five-Number Summaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.06507  2.73445  6.13268  17.38524  462.96690</td>
</tr>
<tr>
<td>R1</td>
<td>0.00679  1.93878  4.41474  10.94515  171.13829</td>
</tr>
<tr>
<td>Rm</td>
<td>0.01317  0.55277  1.13498  2.06179   6.64792</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Experiment 4 ($H_2$, $s_2$) Five-Number Summaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.13057  3.82814  10.47636  43.46417  1408.63960</td>
</tr>
<tr>
<td>R1</td>
<td>0.08702  2.77926  7.71088  29.00329  622.24377</td>
</tr>
<tr>
<td>Rm</td>
<td>0.05414  0.98439  2.52893  8.31241  232.95278</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Experiment 5 ($H_1$, $s_3$) Five-Number Summaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.27110  11.03796  33.19717  213.20916  10538.42589</td>
</tr>
<tr>
<td>R1</td>
<td>0.02806  8.02061  22.51994  125.26577  3903.42831</td>
</tr>
<tr>
<td>Rm</td>
<td>0.05622  2.24305  4.54329  9.15378   66.27180</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Experiment 6 ($H_2$, $s_3$) Five-Number Summaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.44305  10.46696  31.28688  134.61936  3867.72410</td>
</tr>
<tr>
<td>R1</td>
<td>0.29529  7.87447  21.31775  90.27438   3185.27260</td>
</tr>
<tr>
<td>Rm</td>
<td>0.07319  3.34006  8.13794  25.75392  670.91954</td>
</tr>
</tbody>
</table>
5. Numerical Experiments

spread amongst the \( m \) design sites. The regression models, to varying degrees, tradeoff exploitation for exploration. These experiments support a conjecture that in terms of building quadratic models given noisy function evaluations that exploration can be more useful than exploitation, i.e., it might be beneficial to observe the objective at an unobserved design site rather than observing replications at a particular design site.

5.3 General Stochastic Optimization – Quasi-Convex Functions

This section describes the numerical results of experiments minimizing bounded, twice continuously differentiable, quasi-convex functions with observations subject to additive noise. Let \( f : \mathbb{R}^p \rightarrow \mathbb{R} \) denote the response surface. We consider functions of the following form:

\[
f(x; A) = 1 - \exp \left( -\frac{1}{\|A\|} x^T A x \right)
\]

where \( A \) is a \( p \times p \) symmetric and positive definite matrix. We experiment with functions where \( p = 2, 5, \) and \( 10 \). In addition to considering general positive definite matrices, we consider the special cases where \( A \) is the \( p \times p \) identity matrix and where \( A \) is a diagonal matrix with positive diagonal entries. For each test dimension we consider the identity matrix (denoted \( A_{p1} \)), two diagonal matrices (denoted \( A_{p2}^p \) and \( A_{p3}^p \)), and two positive definite matrices (denoted \( A_{p4}^p \) and \( A_{p5}^p \)). The fifteen test matrices used in these experiments are listed in the Appendix of this chapter.

We observe \( Y_{x} = f(x; A) + \epsilon \) where \( \epsilon \sim N(0, \sigma^2) \). We repeat each experiment for two levels of noise, \( \sigma = 0.05 \) and 0.1. We compare the Kiefer-Wolfowitz algorithm and QNSTOP-BASIC. Both optimization algorithms start from two different pre-determined starting points selected for each \( p \). Table 5.7 lists the starting points. The constants \( c_2, c_5, \) and \( c_{10} \) are chosen such that \( \|\theta^{p,1}\| = \|\theta^{p,2}\| \) for each \( p = 2, 5, 10 \).
5. Numerical Experiments

As discussed in Section 5.1, parameterization of the algorithms complicates meaningful comparisons between optimization algorithms. To alleviate this issue we test Kiefer-Wolfowitz and QNSTOP-BASIC using 16 different combinations of parameters. The common parameters between the algorithms are listed in Table 5.8. Additionally, QNSTOP-BASIC used $\gamma = 5$ and $\eta = 0.9d/\gamma$. We performed crude preliminary investigations to identify meaningful ranges of the parameters to be tested.

Each algorithm was given 100 iterations for each problem and choice of parameters. Each experiment was replicated 100 times. For each algorithm we take the location of the final iteration

<table>
<thead>
<tr>
<th>#</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1/6</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1/6</td>
<td>0</td>
<td>0.003</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1/6</td>
<td>0</td>
<td>0.010</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1/6</td>
<td>0</td>
<td>0.030</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1/6</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1/6</td>
<td>0</td>
<td>0.003</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>1/6</td>
<td>0</td>
<td>0.010</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1/6</td>
<td>0</td>
<td>0.030</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>1/6</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1/6</td>
<td>0</td>
<td>0.003</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>1/6</td>
<td>0</td>
<td>0.010</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>1/6</td>
<td>0</td>
<td>0.030</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>1/6</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>1/6</td>
<td>0</td>
<td>0.003</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>1/6</td>
<td>0</td>
<td>0.010</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>1/6</td>
<td>0</td>
<td>0.030</td>
</tr>
</tbody>
</table>

Table 5.8: Sixteen combinations of parameters used on the quasi-convex test problems.
5. Numerical Experiments

and compute a noise-free evaluation of the objective function. We record the mean of the final evaluations over the 100 replications. To control for the choice of parameters we report the mean for the set of parameters that gives the smallest mean for each algorithm. These results are compiled in Tables 5.9-5.11. The boldface indicates the lower mean function evaluation between the two algorithms. The differences are not necessarily statistical significant, however, QNSTOP-BASIC has a smaller mean function evaluation in 46 of the 60 experiments.

<table>
<thead>
<tr>
<th>A</th>
<th>($\sigma = 0.05, \theta_0 = \theta^{2,1}$)</th>
<th>($\sigma = 0.05, \theta_0 = \theta^{2,2}$)</th>
<th>($\sigma = 0.1, \theta_0 = \theta^{2,1}$)</th>
<th>($\sigma = 0.1, \theta_0 = \theta^{2,2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KW</td>
<td>QN</td>
<td>KW</td>
<td>QN</td>
<td>KW</td>
</tr>
<tr>
<td>A_1^5</td>
<td>90.0986</td>
<td>11.2774</td>
<td>340.2453</td>
<td>48.8047</td>
</tr>
<tr>
<td>A_2^5</td>
<td>0.0013</td>
<td>0.0043</td>
<td>8.4316</td>
<td>0.3185</td>
</tr>
<tr>
<td>A_3^5</td>
<td>0.0012</td>
<td>0.0011</td>
<td>0.0200</td>
<td>0.0197</td>
</tr>
<tr>
<td>A_4^5</td>
<td>0.0011</td>
<td>0.0011</td>
<td>0.0182</td>
<td>0.0174</td>
</tr>
<tr>
<td>A_5^5</td>
<td>0.5081</td>
<td>0.2496</td>
<td>4.6142</td>
<td>2.0575</td>
</tr>
</tbody>
</table>

Table 5.9: Mean final objective ($\times 10^{-3}$) on the quasi-convex problems ($p = 2$).

<table>
<thead>
<tr>
<th>A</th>
<th>($\sigma = 0.05, \theta_0 = \theta^{2,1}$)</th>
<th>($\sigma = 0.05, \theta_0 = \theta^{2,2}$)</th>
<th>($\sigma = 0.1, \theta_0 = \theta^{2,1}$)</th>
<th>($\sigma = 0.1, \theta_0 = \theta^{2,2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KW</td>
<td>QN</td>
<td>KW</td>
<td>QN</td>
<td>KW</td>
</tr>
<tr>
<td>A_1^10</td>
<td>0.0027</td>
<td>0.0028</td>
<td>0.0429</td>
<td>0.0433</td>
</tr>
<tr>
<td>A_2^10</td>
<td>0.0026</td>
<td>0.0025</td>
<td>0.0416</td>
<td>0.0406</td>
</tr>
<tr>
<td>A_3^10</td>
<td>0.0026</td>
<td>0.0024</td>
<td>0.0416</td>
<td>0.0408</td>
</tr>
<tr>
<td>A_4^10</td>
<td>0.0051</td>
<td>0.0153</td>
<td>0.0685</td>
<td>0.0489</td>
</tr>
<tr>
<td>A_5^10</td>
<td>0.0025</td>
<td>0.0025</td>
<td>0.0403</td>
<td>0.0383</td>
</tr>
</tbody>
</table>

Table 5.10: Mean final objective ($\times 10^{-3}$) on the quasi-convex problems ($p = 5$).

<table>
<thead>
<tr>
<th>A</th>
<th>($\sigma = 0.05, \theta_0 = \theta^{2,1}$)</th>
<th>($\sigma = 0.05, \theta_0 = \theta^{2,2}$)</th>
<th>($\sigma = 0.1, \theta_0 = \theta^{2,1}$)</th>
<th>($\sigma = 0.1, \theta_0 = \theta^{2,2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KW</td>
<td>QN</td>
<td>KW</td>
<td>QN</td>
<td>KW</td>
</tr>
<tr>
<td>A_1^10</td>
<td>0.0055</td>
<td>0.0053</td>
<td>0.0515</td>
<td>0.0794</td>
</tr>
<tr>
<td>A_2^10</td>
<td>0.0055</td>
<td>0.0051</td>
<td>0.0872</td>
<td>0.0817</td>
</tr>
<tr>
<td>A_3^10</td>
<td>0.0054</td>
<td>0.0089</td>
<td>0.0867</td>
<td>0.0848</td>
</tr>
<tr>
<td>A_4^10</td>
<td>0.0055</td>
<td>0.0051</td>
<td>0.0839</td>
<td>0.0808</td>
</tr>
<tr>
<td>A_5^10</td>
<td>0.0055</td>
<td>0.0049</td>
<td>0.0881</td>
<td>0.0730</td>
</tr>
</tbody>
</table>

Table 5.11: Mean final objective ($\times 10^{-3}$) on the quasi-convex problems ($p = 10$).
5. Numerical Experiments

5.4 Simulation Optimization – Ambulance Base Location

Example 1.1 in Chapter 1 describes the placement of ambulance bases in a city. In this section, we provide numerical results for one such experiment. We represent the city by the unit square $[0,1] \times [0,1]$. There are $b = 5$ ambulance bases to be placed in the square. Emergency calls follow a Poisson process with rate parameter $\lambda_a = 0.5$ and the locations of the calls follow a joint triangle distribution with mode at $(0.8, 0.8)$. The ambulances travel to and from each site at $v = 0.5$ and the time spent at each call site follows an exponential distribution with parameter $\lambda_s = 0.1$. The nearest available ambulance is dispatched to the earliest call.

We simulate the city for $t = 1000$ time units and compute the mean response time (waiting time + one-way commute time). Figure 5.1 shows representative calls with five ambulance bases placed by QNSTOP-BASIC with a budget of 2000 simulations starting from $(0.5, ..., 0.5)$, i.e., all of the bases started in the center of the city.

Let $\theta_{qb}$ denote the configuration of bases output by QNSTOP-BASIC. Let $Y_{\theta_{qb}}$ denote the random variable representing the mean waiting times over $t = 1000$ time units given the configuration $\theta_{qb}$. We observe $Y_{\theta_{qb}}$ 30 times. Deng and Ferris [25] use an adaptation of the DIRECT algorithm to position the ambulance bases. Let $\theta_{df}$ denote the configuration in [25] obtained using a budget of 20,000 simulations. We observe $Y_{\theta_{df}}$ 30 times. The mean observation of $Y_{\theta_{qb}}$ was 0.4307 (sd=0.0196) and the mean observation of $Y_{\theta_{df}}$ was 0.4420 (sd=0.0166).

5.5 Parameter Estimation – Tumor Recurrence

This section describes numerical results for Example 1.2 in Chapter 1. Atkinson et al. [1] described a four parameter model for tumor recurrence. We use the simulation-based parameter
5. Numerical Experiments

Figure 5.1: Representative calls with five ambulance bases.
estimation approach described in Section 1.2.

We generated a sample \((m = 500)\) to represent tumor recurrence observations, but at a known \(\theta\). Because the known parameters are the optimal solution we denote them as \(\theta_{opt}\). Let \(\hat{P}_n(\cdot; x)\) denote the empirical distribution of the simulation sample \((n = 500)\). Section 5.5.1 describes results using the Cramér-von Mises test statistic as a measure of goodness-of-fit and Section 5.5.2 describes results using the Kolmogorov-Smirnov test statistic.

We used the Kiefer-Wolfowitz algorithm and QNSTOP-BASIC to estimate the parameters of the tumor recurrence model. The parameters were scaled by the following:

\[
\begin{align*}
\theta'_1 & \leftarrow 1 \times 10^{-1} \theta_1 \\
\theta'_2 & \leftarrow 1 \times 10^{-3} \theta_2 \\
\theta'_3 & \leftarrow 1 \times 10^{-9} \theta_3 \\
\theta'_4 & \leftarrow 1 \times 10^{-10} \theta_4.
\end{align*}
\]

The decision variables of the optimization problem are the vector \(\theta' = (\theta'_1, \theta'_2, \theta'_3, \theta'_4)^T\). However, to simplify matters we treat these as if they are the original variables \(\theta\).
5. Numerical Experiments

5.5.1 Cramér-von Mises

Each observation of the objective function is computed by drawing a sample \( n = 500 \) from \( P(\cdot; x) \) and computing the plug-in estimator for the objective:

\[
\hat{f}_n(\theta) = \Delta_{cvm}(\hat{P}_n(\cdot; \theta), \hat{Q}_m(\cdot)) = \int_{-\infty}^{\infty} \left( \hat{P}_n(z; \theta) - \hat{Q}_m(z) \right)^2 d\hat{P}_n(z) = \sum_{i=1}^{n} \left( \hat{P}_n(p_i; \theta) - \hat{Q}_m(p_i) \right)^2.
\]

We started each algorithm from \( \theta_0 = (10, 10, 10, 10)^T \) and used the parameters \( a = 4, b = 1/6, c = 10, d = 0.001 \). QNSTOP-BASIC used \( \gamma = 5 \) and \( \eta = 0.9d/\gamma \). Figure 5.2 shows the observed values of the plug-in estimator at \( \{\theta_1, \ldots, \theta_{100}\} \) for each algorithm. We observe 1000 replications of the plug-in estimator at the final estimate obtained by each algorithm. Furthermore, we observe 1000 replications at the known optimal solution. Figure 5.3 plots the empirical distributions of the responses at the estimates found by the Kiefer-Wolfowitz algorithm and QNSTOP-BASIC as well as the responses at the global optimum.

We replicated the optimization runs 100 times. The mean objective value at each iteration is computed for each algorithm and plotted in Figure 5.4.

We repeat the experiment from a different starting point, \( \theta_0 = (0.1, 0.1, 0.1, 0.1)^T \). Like above, Figure 5.5 plots the empirical distributions of the responses at the estimates found by the Kiefer-Wolfowitz algorithm and QNSTOP-BASIC as well as the responses at the global optimum.
Figure 5.2: Estimation of the tumor model parameters using the Cramér-von Mises test statistic to measure goodness-of-fit. Observations at iterates of the Kiefer-Wolfowitz algorithm and QNSTOP-BASIC.
5. Numerical Experiments

Figure 5.3: Estimation of the tumor model parameters using the Cramér-von Mises test statistic to measure goodness-of-fit. Empirical distributions of the response at the optimal solutions.
Figure 5.4: Estimation of the tumor model parameters using the Cramér-von Mises test statistic to measure goodness-of-fit. Mean observations at iterates of the Kiefer-Wolfowitz and QNSTOP-BASIC over 100 replications.
Figure 5.5: Estimation of the tumor model parameters using the Cramér-von Mises test statistic to measure goodness-of-fit. Empirical distributions of the response at the optimal solutions found starting at $\theta_0 = (0.1, 0.1, 0.1, 0.1)^T$. 

\[ 5. \text{ Numerical Experiments} \]
5. Numerical Experiments

Table 5.12: Estimation of the tumor model parameters using the Kolmogorov-Smirnov test statistic to measure goodness-of-fit. Mean and standard deviations of observations at the final estimate of the parameters using the Kiefer-Wolfowitz algorithm and QNSTOP-BASIC.

<table>
<thead>
<tr>
<th>a</th>
<th>d</th>
<th>(\theta_0 = (10, 10, 10)^T)</th>
<th>(\theta_0 = (0.1, 0.1, 0.1, 0.1)^T)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>KW</td>
<td>QN</td>
</tr>
<tr>
<td>4</td>
<td>0.002</td>
<td>0.1234</td>
<td>0.0985</td>
</tr>
<tr>
<td>4</td>
<td>0.003</td>
<td>0.0892</td>
<td>0.0173</td>
</tr>
<tr>
<td>4</td>
<td>0.004</td>
<td><strong>0.0880</strong></td>
<td>0.0139</td>
</tr>
<tr>
<td>4</td>
<td>0.005</td>
<td>0.0984</td>
<td>0.0189</td>
</tr>
<tr>
<td>8</td>
<td>0.002</td>
<td>0.1938</td>
<td>0.0224</td>
</tr>
<tr>
<td>8</td>
<td>0.003</td>
<td>0.1968</td>
<td>0.0202</td>
</tr>
<tr>
<td>8</td>
<td>0.004</td>
<td>0.1980</td>
<td>0.0173</td>
</tr>
<tr>
<td>8</td>
<td>0.005</td>
<td>0.2014</td>
<td>0.0159</td>
</tr>
<tr>
<td>12</td>
<td>0.002</td>
<td>0.2632</td>
<td>0.0172</td>
</tr>
<tr>
<td>12</td>
<td>0.003</td>
<td>0.2634</td>
<td>0.0166</td>
</tr>
<tr>
<td>12</td>
<td>0.004</td>
<td>0.2814</td>
<td>0.0172</td>
</tr>
<tr>
<td>12</td>
<td>0.005</td>
<td>0.2878</td>
<td>0.0142</td>
</tr>
<tr>
<td>16</td>
<td>0.002</td>
<td>0.3486</td>
<td>0.0161</td>
</tr>
<tr>
<td>16</td>
<td>0.003</td>
<td>0.3496</td>
<td>0.0140</td>
</tr>
<tr>
<td>16</td>
<td>0.004</td>
<td>0.3444</td>
<td>0.0124</td>
</tr>
<tr>
<td>16</td>
<td>0.005</td>
<td>0.3492</td>
<td>0.0123</td>
</tr>
</tbody>
</table>

5.5.2 Kolmogorov-Smirnov

In this section, we use the Kolmogorov-Smirnov test statistic to assess goodness-of-fit for estimating the parameters of the tumor model. We repeat the same experiment for sixteen choices of the scaling parameters \((b = 1/6\) and \(c = 1\) for all experiments). We test both algorithms using various choices of the parameters to demonstrate the change in performance due to scaling.

We ran each algorithm 10 times and observed a function evaluation at each estimate of the parameters obtained by each algorithm. Table 5.12 shows the mean and standard deviation of observations at the solutions.

Figure 5.6 plots the empirical distributions of the responses at the estimates found by the Kiefer-Wolfowitz algorithm and QNSTOP-BASIC starting at \(\theta_0 = (10, 10, 10, 10)^T\) as well as the
Figure 5.6: Estimation of the tumor model parameters using the Kolmogorov-Smirnov test statistic to measure goodness-of-fit. Empirical distributions of the response at the optimal solutions found starting at $\theta_0 = (10, 10, 10)^T$.

responses at the global optimum. Figure 5.7 plots the empirical distributions starting at $\theta_0 = (0.1, 0.1, 0.1, 0.1)^T$. 

85
Figure 5.7: Estimation of the tumor model parameters using the Kolmogorov-Smirnov test statistic to measure goodness-of-fit. Empirical distributions of the response at the optimal solutions found starting at $\theta_0 = (0.1, 0.1, 0.1, 0.1)^T$. 
5. Numerical Experiments

Appendix

This section lists the matrices used in Section 5.3. For $p = 2$:

\[
A_2^2 = \text{diag}(1, 1), A_2^3 = \text{diag}(4, 1), A_2^4 = \text{diag}(8, 1),
\]

\[
A_2^2 = \begin{bmatrix}
4 & -2 \\
-2 & 2
\end{bmatrix}, A_2^3 = \begin{bmatrix}
12 & 3 \\
3 & 1
\end{bmatrix}
\]

where the superscripts on each matrix denote the dimension.

For $p = 5$:

\[
A_5^1 = \text{diag}(1, 1, 1, 1, 1), A_5^2 = \text{diag}(8, 4, 2, 1, 1), A_5^3 = \text{diag}(10, 3, 3, 2, 2),
\]

\[
A_5^1 = \begin{bmatrix}
4 & -3 & -2 & -3 & -4 \\
-3 & 14 & -3 & -2 & 1
\end{bmatrix}, A_5^3 = \begin{bmatrix}
4 & 0 & -1 & 1 & 0 \\
0 & 2 & 1 & 0 & 0
\end{bmatrix}
\]

\[
A_5^3 = \begin{bmatrix}
-1 & 1 & 7 & -2 & 0 \\
1 & 0 & -2 & 3 & -1
\end{bmatrix}
\]

For $p = 10$:

\[
A_{10}^1 = \text{diag}(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
\]

\[
A_{10}^2 = \text{diag}(5, 5, 7, 7, 8, 3, 7, 7, 2, 2)
\]

\[
A_{10}^3 = \text{diag}(9, 6, 2, 2, 3, 9, 3, 8, 3, 9)
\]
5. Numerical Experiments

\[
A_{10}^4 = \begin{bmatrix}
6 & 0 & 0 & -1 & 1 & 0 & -1 & -1 & 0 & 1 \\
0 & 7 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 \\
0 & 0 & 6 & 0 & 2 & -1 & 0 & 1 & 0 & 1 \\
-1 & 1 & 0 & 6 & 1 & 0 & 0 & -1 & 0 & 0 \\
1 & 0 & 2 & 1 & 8 & -2 & 0 & 0 & 1 & 1 \\
0 & 0 & -1 & 0 & -2 & 6 & 0 & 0 & 0 & -2 \\
-1 & 0 & 0 & 0 & 0 & 0 & 7 & 1 & 0 & -1 \\
-1 & -1 & 1 & -1 & 0 & 0 & 1 & 8 & 1 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 7 & 1 \\
1 & 0 & 1 & 0 & 1 & -2 & -1 & 0 & 1 & 8 \\
\end{bmatrix}
\]

\[
A_{10}^5 = \begin{bmatrix}
9 & -1 & -2 & -2 & 1 & -1 & 0 & -1 & 1 & 1 \\
-1 & 11 & 0 & 2 & 0 & -1 & -2 & -4 & -5 & 0 \\
-2 & 0 & 10 & 1 & 6 & -3 & 0 & 2 & 1 & 2 \\
-2 & 2 & 1 & 10 & 3 & 1 & -3 & -4 & -2 & 1 \\
1 & 0 & 6 & 3 & 16 & -5 & -4 & -3 & 1 & 4 \\
-1 & -1 & -3 & 1 & -5 & 8 & 1 & 0 & -2 & -5 \\
0 & -2 & 0 & -3 & -4 & 1 & 16 & 8 & 2 & -4 \\
-1 & -4 & 2 & -4 & -3 & 0 & 8 & 18 & 5 & -3 \\
1 & -5 & 1 & -2 & 1 & -2 & 2 & 5 & 10 & 1 \\
1 & 0 & 2 & 1 & 4 & -5 & -4 & -3 & 1 & 13 \\
\end{bmatrix}
\]
Part II

Proximity-Based Methods for
Disparate Information Fusion
6

Feature Extraction for Multiple Kernel Learning

6.1 Introduction

The standard construction of a support vector machine (SVM) requires a kernel (inner product) matrix. In many supervised learning tasks, multiple kernels are available for consideration. Multiple Kernel Learning (MKL) synthesizes a single kernel from a set of multiple kernels for the purpose of constructing an SVM. To date, most techniques for MKL have formed linear (often convex) combinations of the given kernels.

Kernels provided by domain experts are typically of high rank, corresponding to high-dimensional feature spaces. It is widely appreciated that performance on a supervised learning task in a high-dimensional feature space may benefit from preliminary feature selection or extraction. This is a form of regularization: the idea is to use only essential information in constructing the classifier. Stated another way, statistical models of many dimensions often have low bias but high variance. Dimension reduction may substantially decrease variance while only slightly increasing
bias.

A variety of methods exist for supervised feature extraction. Although the rationale for extracting useful low-rank kernels from high-rank kernels is identical, less is known about how to do so. We propose Multiple Kernel Basis Extraction (MKBE), a technique by which the original high-dimensional kernels are replaced by a number of rank-one kernels. A standard MKL technique is then applied to the rank-one kernels to construct an SVM.

6.2 Preliminaries

Let $S = \{(x_j, y_j)\}_{j=1}^{N}$ be a set of learning instances, where $x_j \in \mathcal{X}$ and $y_j \in \{-1, 1\}$ is the label for $x_j$. By a classifier, we mean a function $f : \mathcal{X} \rightarrow \mathbb{R}$ with the interpretation that $f$ assigns label 1 to $x$ if $f(x) > 0$ and label $-1$ if $f(x) < 0$.

Assume that we are given a collection of $p$ kernels, $\mathcal{K} = \{K_1, \ldots, K_p\}$. A common approach to MKL is to construct a classifier of the form

$$f(x_i) = \sum_{j=1}^{N} \alpha_j^* K_{ij}^* + b^*, \quad (6.1)$$

where $K^* = [K_{ij}^*]$ is a convex combination of the $K_i \in \mathcal{K}$, i.e.,

$$K^* = \sum_{i=1}^{p} \mu_i K_i, \quad \mu_i \geq 0, \quad \sum_{i=1}^{p} \mu_i = 1. \quad (6.2)$$

Such classifiers can be constructed in a variety of ways. Lanckriet et al. [49] posed the problem as a quadratically constrained quadratic program. Bach et al. [2] showed that the problem formulated in [49] is equivalent to a mixed $(l_2, l_1)$ norm regularization on the weight vector and proposed an SMO-like algorithm to solve it. Sonnenburg et al. [76] formulated the problem as a semi-infinite
linear program (SILP). Their algorithm is an instance of a class of exchange methods for solving SILPs. Rakotomamonjy et al. [68] used $l_2$ norm regularization on the weight vector and proposed an algorithm based on gradient descent.

All of the above approaches manipulate entire kernels. In contrast, one might write each kernel as the sum of rank-one kernels, $K_i = \sum_{j=1}^{N} M_{ij}$, then apply any of the above approaches to (a subset of) the collection of rank-one kernels, $\mathcal{M} = \{M_{ij}\}$. This is a form of feature extraction. If most of the discriminating information contained in the $K_i$ can be represented by a small number of $M_{ij}$, then we anticipate that this procedure will produce better classifiers than procedures based on convex combinations of the $K_i$.

An obvious way to proceed is by spectral decomposition, i.e., by writing

$$K_i = \sum_{j=1}^{N} \lambda_{ij} \nu_{ij} \nu_{ij}^T,$$

(6.3)

where $\lambda_{ij}$ and $\nu_{ij}$ are the eigenvalues and corresponding eigenvectors of $K_i$. Combining (6.2) and (6.3), we obtain the representation

$$K^* = \sum_{i=1}^{P} \sum_{j=1}^{N} \mu_i \lambda_{ij} \nu_{ij} \nu_{ij}^T.$$

Typically, $NP$ will be large and there will be too many rank-one kernels for standard MKL algorithms to use all of them. One can limit the number of eigenvectors extracted from each $K_i$, but it won’t do to simply select the eigenvectors that correspond to the largest eigenvalues: if some of the $K_i$ contain similar information (for example, multiple Gaussian kernels with slightly different bandwidths are sometimes used in computer vision [90, 80]), then the eigenvectors extracted from these $K_i$ may be nearly identical. To avoid this difficulty, we propose constructing rank-one kernels from a single set of orthogonal vectors. The following section explains how these vectors are
6. Feature Extraction for Multiple Kernel Learning

extracted.

6.3 Multiple Kernel Basis Extraction

We seek to extract \( \mathcal{V} = \{v_1, \ldots, v_m\} \), a set of orthogonal vectors, from \( \mathcal{K} \), a set of kernels. To ensure that these vectors will have predictive value, we rely on the notion of kernel alignment. The alignment between kernels \( K_1 \) and \( K_2 \) is

\[
A(K_1, K_2) = \frac{\text{tr}(K_1 K_2)}{\sqrt{\text{tr}(K_1 K_1) \text{tr}(K_2 K_2)}},
\]

the cosine of the angle between \( K_1 \) and \( K_2 \). This quantity was used for kernel selection in [21]. We might use other criteria, but it turns out that maximizing alignment reduces to simple calculations for maximizing Rayleigh quotients.

We divide the set of learning instances, \( \mathcal{S} \), into \( \mathcal{S}_{\text{train}} \) and \( \mathcal{S}_{\text{test}} \), then use \( \mathcal{S}_{\text{train}} \) to supervise the construction of \( \mathcal{V} \). Let \( N_{\text{train}} = |\mathcal{S}_{\text{train}}| \). Given \( x \in \mathbb{R}^N \), we denote the components of \( x \) that correspond to elements of \( \mathcal{S}_{\text{train}} \) by \( \hat{x} \in \mathbb{R}^{N_{\text{train}}} \).

We construct \( \mathcal{V} \) sequentially. Initially, compute

\[
q_{1,i} = \operatorname{argmax}_{x \neq 0} \frac{x^T K_i x}{x^T x}
\]

for \( i = 1, \ldots, p \), then set

\[
v_1 = \operatorname{argmax}_{q_{1,i}} A \left( \begin{bmatrix} q_{1,i} & q_{1,i}^T \end{bmatrix}, [\hat{y} \hat{y}^T] \right).
\]
Subsequently, given $v_1, \ldots, v_k$, compute

$$ q_{k+1,i} = \arg\max_{x \neq 0, x \perp v_1, \ldots, v_k} \frac{x^T K_i x}{x^T x}, $$

then set

$$ v_{k+1} = \arg\max_{q_{k+1,i}} A \left( [\hat{q}_{k+1,i}, \hat{q}_{k+1,i}^T], [\hat{y} \hat{y}^T] \right). $$

Notice that the alignment between rank-one kernels $\hat{q}^T \hat{q}$ and $\hat{y} \hat{y}^T$ is just a Raleigh quotient:

$$ A (\hat{q} \hat{q}^T, \hat{y} \hat{y}^T) = \frac{\text{tr} (\hat{q} \hat{q}^T \hat{y} \hat{y}^T)}{\sqrt{\text{tr} (\hat{q} \hat{q}^T \hat{q} \hat{q}^T)} \sqrt{\text{tr} (\hat{y} \hat{y}^T \hat{y} \hat{y}^T)}} = \frac{1}{N_{\text{train}}} \frac{\hat{q}^T (\hat{y} \hat{y}^T) \hat{q}}{\hat{q}^T \hat{q}}. $$

Furthermore, if $C_k$ denotes the $k \times N$ matrix with rows $v_1^T, \ldots, v_k^T$, then the problem of computing $q_{k+1,i}$ is a constrained Rayleigh quotient problem with constraints $C_k x = 0$ [34]. The constrained problem can be written as an unconstrained problem,

$$ q_{k+1,i} = \arg\max_{x \neq 0} \frac{x^T P_k K_i P_k x}{x^T x}, $$

where $P_k$ is the projection matrix $I_N - C_k^T (C_k C_k^T)^{-1} C_k$. Because the rows of $C_k$ are orthogonal, $C_k C_k^T = I_k$ and $P_k = I_N - C_k^T C_k$. Thus, each step of our procedure for constructing $V$ involves maximizing a Rayleigh quotient. The entire algorithm, which has a worst case computational complexity of $O(N^3 p m)$, is summarized below.
Algorithm 6.1. Multiple Kernel Basis Extraction (MKBE)

0. Initialize $V \leftarrow \emptyset$ and $P \leftarrow I_N$

1. For $k \leftarrow 1$ to $m$
   
   (a) $Q \leftarrow \emptyset$
   
   (b) For $K_i \in K$
      
      i. $q \leftarrow$ eigenvector associated with largest eigenvalue of $PK_iP$
      
      ii. $Q \leftarrow Q \cup \{q\}$
   
   (c) $v_k \leftarrow \text{argmax}_{q \in Q} A(\hat{q}^{\top}, \hat{y}^{\top})$
   
   (d) $V \leftarrow V \cup \{v_k\}$
   
   (e) $C_k \leftarrow k \times N$ matrix with rows $v_1^{\top}, \ldots, v_k^{\top}$
   
   (f) $P \leftarrow I_N - C_kC_k^{\top}$

6.4 Error Bounds for Transduction

Following Section 5 of [49], we consider the problem of transduction, i.e., of classifying the test data using a classifier constructed from the training data.

The SVM $f$ in (6.1) classifies $x_i$ according to the sign of $f(x_i)$, with $f(x_i) = \pm 1$ if $x_i$ is a support vector. The vector $x_i$ is misclassified if $y_i f(x_i) \leq 0$. Consider two loss functions,

$$L_1 (f(x_i), y_i) = \begin{cases} 
1 & y_i f(x_i) \leq 0 \\
0 & y_i f(x_i) > 0 
\end{cases}$$
6. Feature Extraction for Multiple Kernel Learning

and

\[ L_2 (f(x_i), y_i) = \begin{cases} 
1 - y_i f(x_i) & y_i f(x_i) \leq 1 \\
0 & y_i f(x_i) > 1 
\end{cases} \]

The former allows us to count misclassification errors; the latter is a margin cost function.

For convenience, assume that \( N_{\text{train}} = N/2 = n \). We measure the performance of \( f \) on \( S_{\text{test}} \) using \( L_1 \) and the performance of \( f \) on \( S_{\text{train}} \) using \( L_2 \). Let

\[ er(f) = \frac{1}{n} \sum_{i \in S_{\text{test}}} L_1 (f(x_i), y_i), \]

the proportion of misclassification errors in the test set, and

\[ mc(f) = \frac{1}{n} \sum_{i \in S_{\text{train}}} L_2 (f(x_i), y_i), \]

the average cost of margin violation in the training set. We interpret the difference, \( \text{gen}(f) = er(f) - mc(f) \), as a measure of generalization.

Because \( L_1 (f(x_i), y_i) \leq L_2 (f(x_i), y_i) \), classifiers for which \( \text{gen}(f) \gg 0 \) have poor generalization.

To remove the dependence of \( \text{gen}(f) \) on how \( S \) is partitioned into \( S_{\text{train}} \) and \( S_{\text{test}} \), we assume a probability model under which each of the \( \binom{N}{n} \) possible partitions is equally likely. One can then measure how well \( f \) generalizes by considering the probability that \( \text{gen}(f) \) is small.

Consider kernel classifiers with margin \( \gamma \) of the form

\[ F_K(\gamma) = \left\{ x_j \mapsto \sum_{i=1}^{2n} \alpha_i K_{ij} : K \in \mathcal{K}, \alpha^T K \alpha \leq \gamma^{-2} \right\}. \]

Using a proof technique introduced by Koltchinskii and Panchenko [44], Lanckriet et al. [49, Theorem 24] derived the following bound.
Proposition 6.1. Fix $\gamma > 0$ and $\delta \in (0, 1)$. Assume that each partition of $S$ into $S_{\text{train}}$ and $S_{\text{test}}$ is equally likely. Then every $f \in \mathcal{F}_K(\gamma)$ satisfies

$$P\left(\text{gen}(f) \leq \frac{4 + \sqrt{-2\log\delta}}{\sqrt{n}} + \frac{\sqrt{\mathcal{C}(K)}}{n\gamma} \right) \geq 1 - \delta,$$

where

$$\mathcal{C}(K) = \mathbb{E}_{\sigma \in \Delta} \sigma^\top K \sigma$$

and the expectation is with respect to $\sigma$ drawn uniformly from $\{\pm 1\}^N$.

Lanckriet et al. refer to $\mathcal{C}(K)$ as the complexity of $K$. Notice that, if $K_1 \subseteq K_2$, then $\mathcal{C}(K_1) \leq \mathcal{C}(K_2)$.

Furthermore, for $K = \{K_1, \ldots, K_m\}$, let

$$K_c^+ = \left\{ K = \sum_{j=1}^m \mu_j K_j : K \succeq 0, \; \mu_j \geq 0, \; \text{tr}(K) \leq c \right\}.$$

Lanckriet et al. [49, Theorem 24] also established that

$$\mathcal{C}(K_c^+) \leq c \cdot \min \left( m, \frac{n \cdot \max_j \lambda_j}{\text{tr}(K_j)} \right),$$

where $\lambda_j$ is the largest eigenvalue of $K_j$.

Our concern is with multiple kernel learning from the collection of extracted kernels, $\mathcal{M} = \{M_1, \ldots, M_m\}$, where $M_j = v_j v_j^\top$ and therefore $\text{rank}(M_j) = 1$. Accordingly, we apply Proposition 6.1 to the set of kernels $\text{conv}(\mathcal{M})$, the convex hull of $\mathcal{M}$. To bound

$$\mathcal{C}(\text{conv}(\mathcal{M})) = \mathbb{E}_{M \in \text{conv}(\mathcal{M})} \max_{\mu \in \Delta} \sigma^\top M \sigma = \mathbb{E}_{\mu \in \Delta} \sum_{j=1}^m \mu_j \sigma^\top M_j \sigma,$$
where \( \Delta \) denotes the unit simplex in \( \mathbb{R}^m \), let

\[
\begin{align*}
  c &= \max_{M \in \text{conv}(M)} \text{tr}(M) = \max_j \text{tr}(M_j) \\
  &= \max_j \text{tr}(v_jv_j^T) = \max_j \text{tr}(v_j^Tv_j) = 1.
\end{align*}
\]

Then \( \text{conv}(M) \subseteq M_+^+ \), so \( C(\text{conv}(M)) \leq C(M_+^+) \). Furthermore, because \( \text{rank}(M_j) = 1 \), \( \lambda_j = \text{tr}(M_j) \) and the complexity bound simplifies to

\[
C(\text{conv}(M)) \leq \min(m, n).
\]

Combining the above results, we see that extracting a small number \( m \) of rank-one kernels leads to a kernel classifier that should have good generalization. Hence, when the predictive information in \( K = \{K_1, \ldots, K_p\} \) can be captured by a small number of rank-one kernels, we would expect MKL based on \( M = \{M_1, \ldots, M_m\} \) to outperform MKL based on \( K = \{K_1, \ldots, K_p\} \).

### 6.5 Numerical Experiments

We demonstrate MKBE on several data sets, using SimpleMKL [68] to learn from the extracted rank-one kernels.

#### 6.5.1 Stars

Sonnenburg et al. [76] considered the problem of discriminating between two concentric star-shaped distributions. Following their lead, we performed four experiments, fixing the radius of the inner star at 4 and varying the radius of the outer star in \( \{5, 6, 7, 8\} \). For each experiment, we
6. Feature Extraction for Multiple Kernel Learning

generated 1000 points from each distribution. Scatter plots of the resulting data sets are displayed in Figure 6.1.

We randomly divided each data set into training and testing sets of $n = 1000$ points and constructed seven different Gaussian kernels, $\{K_1, \ldots, K_7\}$, defined by

$$K_\sigma = \frac{\exp \left( -\frac{||x_i - x_j||^2}{2\sigma^2} \right)}{2\sigma^2}.$$

We then constructed kernel classifiers by two methods. The first method applied SimpleMKL directly to $\{K_1, \ldots, K_7\}$. The second method used MKBE to extract $\{v_1, \ldots, v_{10}\}$, then applied
SimpleMKL to \{v_1v_1^T, \ldots, v_{10}v_{10}^T\}. (To conserve memory, we modified the Matlab implementation of SimpleMKL to operate directly on \{v_1, \ldots, v_{10}\}.) We experimented with several choices of SimpleMKL’s SVM hyperparameter \(C\) and discovered that varying it did not greatly affect performance. All of the results reported below were obtained using \(C = 100\).

Classifier performance was measured by area under the ROC curve. Results are displayed in Table 6.1. For outer radii of 6, 7, 8, replacing the seven Gaussian kernels with the ten rank-one kernels extracted by MKBE resulted in substantially improved classifier performance. Classifier performance did not improve for an outer radius of 5. This is the most difficult case, and it may be that more than \(m = 10\) rank-one kernels are needed to extract sufficient information from the Gaussian kernels.

<table>
<thead>
<tr>
<th>Outer Radius</th>
<th>7 Gaussian Kernels</th>
<th>10 MKBE Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.60</td>
<td>0.55</td>
</tr>
<tr>
<td>6</td>
<td>0.59</td>
<td>0.79</td>
</tr>
<tr>
<td>7</td>
<td>0.85</td>
<td>0.95</td>
</tr>
<tr>
<td>8</td>
<td>0.75</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Table 6.1: Star Experiments. Area under the ROC curve for two sets of kernels, seven Gaussian kernels and ten rank-one kernels extracted from them by MKBE. Multiple kernel learning was performed by SimpleMKL.

### 6.5.2 Handwritten Digit Recognition

Using the MNIST database of handwritten digits (obtained from http://yann.lecun.com/exdb/mnist/) we attempted to discriminate between the digits 1 and 7. We randomly selected 1000 examples of each digit, then randomly divided these \(N = 2000\) digits into training and testing sets of \(n = 1000\) digits. We constructed \(p = 10\) kernels, five Gaussian with \(\sigma = 0.01, 0.1, 1, 10, 100\), and five polynomial, i.e.,

\[
K_d = \left[ (x_i^T x_j + 1)^d \right]_{ij}
\]
with degree $d = 2, 3, 4, 5, 6$.

As above, we constructed kernel classifiers by two methods. The first method applied SimpleMKL directly to $\{K_1, \ldots, K_{10}\}$. The second method used MKBE to extract $\{v_1, \ldots, v_{10}\}$, then applied SimpleMKL to $\{v_1 \mathbf{v}_1^T, \ldots, v_{10} \mathbf{v}_{10}^T\}$. In this experiment, classifier performance varied with the choice $C$, the SVM hyperparameter in SimpleMLK.

Again, performance was measured by area under the ROC curve. Results are displayed in Table 6.2. The best performance was obtained by applying SimpleMKL with $C \leq 1$ to $\{v_1 \mathbf{v}_1^T, \ldots, v_{10} \mathbf{v}_{10}^T\}$.

<table>
<thead>
<tr>
<th>$C$</th>
<th>10 Original Kernels</th>
<th>10 MKBE Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.54</td>
<td>0.90</td>
</tr>
<tr>
<td>0.5</td>
<td>0.57</td>
<td>0.90</td>
</tr>
<tr>
<td>1</td>
<td>0.56</td>
<td>0.89</td>
</tr>
<tr>
<td>10</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>100</td>
<td>0.63</td>
<td>0.65</td>
</tr>
<tr>
<td>200</td>
<td>0.57</td>
<td>0.56</td>
</tr>
</tbody>
</table>

Table 6.2: Digit Recognition ($\mathbf{1 vs 7}$) Experiments. Area under the ROC curve for two sets of kernels, ten kernels (five Gaussian, five polynomial) and ten rank-one kernels extracted from them by MKBE. Multiple kernel learning was performed by SimpleMKL with various choices of the SVM hyperparameter $C$.

### 6.6 Discussion

Performance on supervised learning tasks in high-dimensional feature spaces often benefits from preliminary feature extraction. We have proposed a technique (MKBE) that extracts rank-one kernels from high-rank kernels. The extracted kernels are formed from orthogonal vectors that are maximally aligned with the class labels. One then applies standard techniques for multiple kernel learning (MKL) to the rank-one kernels to construct a support vector machine (SVM).

The potential benefits of preliminary feature extraction are based on general statistical considerations. The theory presented in Section 6.4 hints that MKBE may improve the performance of SVMs
constructed by MKL. Section 6.5 presents some examples for which the improvement is fairly dramatic. We believe that these examples provide a provocative proof-of-concept.

The ideas that we have explored have much in common with the ideas explored by Ma et al. [55], who considered the problem of learning from multiple dissimilarity matrices, \( \{ \Delta_1, \ldots, \Delta_p \} \). Ma et al. first embedded each \( \Delta_i \) by classical multidimensional scaling (CMDS). CMDS exploits a well-known connection between squared Euclidean distances and Euclidean inner products. The matrix of squared dissimilarities is subjected to a linear transformation, resulting in a matrix, \( \mathbf{B} \), of fallible inner products. The matrix \( \mathbf{B} \) is then replaced by the inner product matrix, \( \bar{\mathbf{B}} \), that is nearest (in the sense of Frobenius norm) \( \mathbf{B} \), subject to a rank restriction. The inner product matrices \( \{ \bar{\mathbf{B}}_1, \ldots, \bar{\mathbf{B}}_p \} \) are our kernel matrices \( \{ \mathbf{K}_1, \ldots, \mathbf{K}_p \} \).

Our MKBE method of feature extraction differs slightly from Ma et al.’s \( J \)-function approach. The latter forms the Cartesian product of the \( p \) representations constructed by CMDS, performs a principal component analysis on the product, and selects the principal components with the largest values of the \( J \)-statistic described below. Given univariate samples \( \mathbf{u} = (u_1, \ldots, u_r)^T \) and \( \mathbf{v} = (v_1, \ldots, v_s)^T \), let \( \mathbf{q}^T = (\mathbf{u}^T | \mathbf{v}^T) \) and let \( \mathbf{y}^T = (\mathbf{e}_r^T | -\mathbf{e}_s^T) \), where \( \mathbf{e}_r \in \mathbb{R}^r \) and \( \mathbf{e}_s \in \mathbb{R}^s \) are vectors with unit entries. Assume that \( \mathbf{u} \mathbf{e}_r^T + \mathbf{v} \mathbf{e}_s^T = 0 \). Then \( J = |\bar{\mathbf{u}} - \bar{\mathbf{v}}|/\lambda \), where

\[
\lambda^2 = \sum_{i=1}^r u_i^2 + \sum_{j=1}^s v_j^2,
\]

whereas

\[
\text{tr} (\mathbf{q}^T \mathbf{y} \mathbf{y}^T) = \text{tr} (\mathbf{q}^T \mathbf{y} \mathbf{y}^T \mathbf{q}) = (\mathbf{q}^T \mathbf{y})^2 = \left( \sum_{i=1}^r u_i - \sum_{j=1}^s v_j \right)^2,
\]

\[
\text{tr} (\mathbf{q} \mathbf{q}^T) = \text{tr} (\mathbf{q}^T \mathbf{q}) = \sum_{i=1}^r u_i^2 + \sum_{j=1}^s v_j^2 = \lambda^2,
\]

\[
\text{tr} (\mathbf{y} \mathbf{y}^T) = \text{tr} (\mathbf{y}^T \mathbf{y}) = r + s.
\]
and therefore the alignment statistic

\[
A(qq^T, yy^T) = \frac{\left(\sum_{i=1}^{r} u_i - \sum_{j=1}^{s} v_j\right)^2}{\sqrt{(r + s)\lambda^2}}.
\]

Furthermore, Ma et al. select a subset of the principal components of the product representation, whereas our \(v_1\) is the first principal component of one of the individual kernels. Finally, we construct an SVM by MKL, whereas Ma et al. perform a linear discriminant analysis using the selected principal components.
Combining Dissimilarities for Nearest Neighbor Classification

7.1 Introduction

Many classification techniques use measures of dissimilarity (or similarity) to learn a function for predicting the class label of a target object. The $k$-nearest neighbor ($k$-NN) classifier labels a target object by plurality vote of the labels of the $k$ objects nearest the target object. Which objects are deemed nearest, hence the performance of the $k$-NN classifier, evidently depends on the measure of dissimilarity. We often have several reasonable measures of dissimilarity at our disposal. For example, in image retrieval there are numerous useful measures of dissimilarity quantifying differences in color, texture, brightness, and shapes in a pair of images. However, it may not be obvious which measure is best suited for the $k$-NN classifier. One approach to this problem is to select the measure resulting in the lowest cross-validated classification error rate. Another approach, so-called ensemble methods, combine the results of classifiers constructed using each dissimilarity measure separately. The present chapter proposes an alternative approach to the problem. We seek
to learn a new measure of dissimilarity as a function of our original dissimilarities and use it to
determine the $k$-nearest neighbors for the $k$-NN classifier.

We work with dissimilarity measures because of their generality. Specifically, we are interested in
dissimilarity functions that generalize a notion of distance. Such functions arise in a number of situations, e.g., city-block distance between finite vectors, Levenshtein edit distance between sequences, and the shortest path distance between two nodes in a graph. Some dissimilarity functions satisfy the strict requirements of a distance function, e.g., Euclidean distance, Mahalanobis distance, and resistance distance between two nodes in a graph.

We propose CODI-KNN (combining dissimilarities for $k$-NN), a method for learning a new measure of dissimilarity from a set of dissimilarities. Before describing and analyzing CODI-KNN, we put our work in the context of some recent work.

### 7.1.1 Embedding Methods

Embedding methods (see Ma and Priebe[56] and Lanckriet et al. [49]) start by embedding dissimilarities or similarities into a Euclidean or inner product representation. Next, they form the Cartesian product of the individual feature representations and perform the desired analysis using the resulting representation. The approach in [56] constructs a separate Euclidean representation using each dissimilarity matrix by multidimensional scaling, forms the Cartesian product of the Euclidean representations, and subsequently performs feature selection and constructs a linear classifier using the resultant representation. Multiple Kernel Learning (MKL) (see [49]) learns convex combinations of inner product matrices for use in a Support Vector Machine (SVM). In contrast with the [56] approach, MKL forms the Cartesian product of scaled feature representations and constructs a linear classifier using the resultant representation.
7. Combining Dissimilarities for Nearest Neighbor Classification

7.1.2 Combining Dissimilarity Representations

Pekalska and Duin [64, Chapter 10] considers combining so-called Dissimilarity Representations, i.e., vector representations where each feature is the dissimilarity from one of a small set of prototypical objects. One approach is to form the Cartesian product of several such representations, however, the number of dimensions may become unwieldy. An approach proposed in [64] combines dissimilarity representations via simple operations (addition, multiplication, minimum, maximum) and obtains improved performance over using the individual Dissimilarity Representations. These methods benefit from low-dimensional representations and simple calculations, however, they may suffer from discarding useful information in the dissimilarity matrices.

7.1.3 Distance Metric Learning

Distance Metric Learning (DML) (see [93], [92], [39]) often concerns learning a metric to improve clustering or \( k \)-nearest neighbor performance given data in a finite vector representation. Alternatively, one could view it as learning a linear transformation of the feature data such that Euclidean distances in the transformed representation improve the \( k \)-nearest neighbor classifier.

This report concerns combining dissimilarities without an intermediate feature representation. We consider learning a new dissimilarity measure as a function of the original dissimilarities for classification using the \( k \)-NN classifier. The proposed methodology, CODI-KNN, is based on DML formulations.
7. Combining Dissimilarities for Nearest Neighbor Classification

7.2 Combining Dissimilarities

Let \( S = \{(x_i, y_i)\}_{i=1}^{N} \) be a set of iid learning instances where \( x_i \in \Xi \) is a feature representation and \( y_i \in Y = \{1, \ldots, K\} \) is a class label. We seek a classifier, i.e., a function \( g \) that assigns a class label \( y \in Y \) to an unlabeled \( x \in \Xi \). The representation space \( \Xi \) is the Cartesian product of several, possibly heterogeneous, representation spaces, i.e., \( \Xi = \Xi_1 \times \Xi_2 \times \ldots \times \Xi_M \). Typically, each \( \Xi_\ell = \mathbb{R}^{d_\ell} \), however, we provide the flexibility to have different types of feature spaces, e.g., graphs.

We further suppose that each feature space is equipped with a measure of dissimilarity, i.e., \( \delta_\ell : \Xi_\ell \times \Xi_\ell \rightarrow \mathbb{R}_{\geq} \), that measures the difference between a pair of objects with respect to their representation in \( \Xi_\ell \). For instance, in image analysis we may have two dissimilarity functions: one to measure dissimilarity with respect to color and one to measure dissimilarity with respect to texture. Dissimilarity functions satisfy the following properties:

- \( \delta(x_i, x_j) \geq 0 \) (non-negativity),
- \( \delta(x_i, x_i) = 0 \) (reflexivity),
- \( \delta(x_i, x_j) = \delta(x_j, x_i) \) (symmetry).

Some authors refer to \( (\Xi_\ell, \delta_\ell) \) as a premetric space; they are distinct from a metric space by not requiring the following properties:

- \( \delta(x_i, x_j) \leq \delta(x_i, x_k) + \delta(x_k, x_j) \) (triangle inequality),
- \( \delta(x_i, x_j) = 0 \) iff \( x_i = x_j \) (identifiability).

We organize dissimilarities into \( M \) dissimilarity matrices, i.e., \( \Delta_\ell = [\delta_\ell(x_i, x_j)]_{ij} \) for \( \ell = 1, \ldots, M \).
7. Combining Dissimilarities for Nearest Neighbor Classification

Dissimilarities may be used to determine neighbors, i.e., the objects with the smallest dissimilarities from a target object. We denote the set of the $k$ nearest neighbors of an object $x$ with respect to $\delta$ as $N_k^*(x; \delta)$. Furthermore, for each target object $x$ we similarly denote the $k$ nearest within-class neighbors as $N_k^W(x; \delta)$ and the $k$ nearest between-class neighbors as $N_k^B(x; \delta)$.

7.2.1 $k$-Nearest Neighbor Classifier

The $k$-nearest neighbor ($k$-NN) classifier labels instances by majority vote of the $k$ nearest labeled examples. Typically, the nearest neighbors are determined by the learning examples with the smallest distance (e.g., Euclidean), however, we use a dissimilarity function $\delta$ to determine the $k$ nearest neighbors. The classifier thus labels an unlabeled object $x$ according to the rule:

$$g(x) = \arg\max_{y \in Y} \sum_{x_i \in N_k^*(x; \delta)} \mathbb{I}(y_i = y)$$

where $\mathbb{I}(\cdot)$ is the indicator function. Hence, a good dissimilarity function makes within-class dissimilarities smaller than between-class dissimilarities.

7.2.2 Combining Dissimilarities for $k$-Nearest Neighbors

Let $\mu_{ij} := \mu(x_i, x_j) = (\delta_1(x_i, x_j), \ldots, \delta_M(x_i, x_j))^T$ be the $M$-vector of pairwise dissimilarities between the objects indexed by $i$ and $j$. We seek a function of $\mu$ denoted $\delta_* : \mathbb{R}_+^M \to \mathbb{R}$ suited for $k$-NN classification. It is not necessary for $\delta_*$ to be nonnegative, that is to say, it need not be a dissimilarity measure itself.

Consider the case in which the input dissimilarities are Euclidean distances. A natural way to combine these dissimilarities is to form a convex combination of their squares resulting in a squared
Combining Dissimilarities for Nearest Neighbor Classification

(weighted) Euclidean distance. However, we can generalize this idea and learn a heterogeneous polynomial of degree two which we can write in matrix-vector form:

$$\delta^2_{ij}(\mu_{ij}; b, B) = b^T \mu_{ij} + \frac{1}{2} \mu_{ij}^T B \mu_{ij}. \tag{7.1}$$

where $b \in \mathbb{R}^M$ and $B \in \mathbb{R}^{M \times M}$ is symmetric. Let $C_M$ denote the cone of $M \times M$ copositive matrices, i.e.,

$$C_M = \{ A \in \mathbb{R}^{M \times M} : A = A^T, v^T A v \geq 0 \text{ for all } v \in \mathbb{R}^M_+ \}.$$

If the matrix

$$A = \begin{bmatrix} 0 & b^T \\ b & B \end{bmatrix}$$

is copositive (i.e., $A \in C_M$), then $\delta_*$ is a nonnegative function and, hence, a dissimilarity function. However, such a restriction on the coefficients in $b$ and $B$ leads to an intractable optimization problem. In fact, simply checking whether a matrix is copositive is co-NP hard. To simplify matters we still refer to $\delta_*$ as a dissimilarity function despite the fact that we will not restrict it be nonnegative.

Ideally, a dissimilarity function $\delta$ used for $k$-NN defines nearest neighbors such that the majority of the nearest neighbors of a target object are within-class. I.e., for a given object $x_i$ we would like to satisfy the inequality:

$$\sum_{x_j \in N_k^W(x_i; \delta)} \mathbb{I}(x_j \in N_k^W(x_i; \delta)) > \sum_{x_j \in N_k^B(x_i; \delta)} \mathbb{I}(x_j \in N_k^B(x_i; \delta)).$$

Learning a dissimilarity function that minimally violates the above inequality is intractable. In lieu of satisfying the inequality above, we instead seek a dissimilarity function such that within-class neighbors tend to be nearer than between-class neighbors. To learn such a dissimilarity function we use a formulation reminiscent of the Distance Metric Learning formulation in [39].
First, we choose a dissimilarity function to determine the within-class and between-class nearest neighbor graphs. For example, one could use the sum of the squared input dissimilarities:

$$\delta_{SS}(x_i, x_j) = \sum_{\ell=1}^{M} (\delta_{\ell}(x_i, x_j))^2 = \mu_{ij}^T \mu_{ij}. $$

Consider the triples in the set:

$$T = \{(i, j, k) : x_j \in N^W_k(x_i; \delta_{SS}), x_k \in N^B_k(x_i; \delta_{SS}), i = 1, \ldots, N\}.$$ 

We would like a dissimilarity function $\delta_*$ such that for each triple $(i, j, k)$:

$$\delta_*(\mu_{ij}; b, B) < \delta_*(\mu_{ik}; b, B).$$

A tractable loss function towards this end is:

$$L(\mu_{ij}, \mu_{ik}; b, B) = [1 + \delta_*(\mu_{ij}; b, B) - \delta_*(\mu_{ik}; b, B)]_+$$

where $[\cdot]_+$ denotes the positive part of its argument. We propose solving the following regularized optimization problem:

$$\min \frac{1}{|T|} \sum_{(i,j,k) \in T} L(\mu_{ij}, \mu_{jk}; b, B) + \frac{\lambda}{2} \left( \|b\|^2 + \|\text{vech}(B)\|^2 \right)$$

s.t. $b \in \mathbb{R}^M$

$$B = B^T \in \mathbb{R}^{M \times M}$$

where $\lambda$ is a regularization parameter and $\text{vech}(\cdot)$ denotes the half-vectorization of a matrix, i.e., the vectorization of the lower triangular part of the matrix.
In this section, we describe an algorithm to solve (7.2). The optimization problem is convex, however, the use of the hinge loss means it is not continuously differentiable. To get around this complication we use a projected subgradient method. If \( k \) and \( N \) are large one might use the approach to minimize the primal form of the soft-margin SVM in [74]. In this report, however, we assume \( k \) and \( N \) are small enough that computing a subgradient is tractable (computation of the subgradient and the objective are \( O(k^2 N) \)).

First, we re-parameterize the objective function in terms of a single vector of the coefficients of \( b \) and \( B \). Let

\[
\beta = \begin{bmatrix} b \\ \text{vech}(B) \end{bmatrix}
\]

and

\[
\nu_{ij} = \begin{bmatrix} \mu_{ij} \\ \text{vech} \left( \mu_{ij} \mu^T_{ij} - \frac{1}{2} (\mu_{ij} \mu^T_{ij})_{d_8} \right) \end{bmatrix}
\]

where \((\cdot)_{d_8}\) is the diagonal matrix obtained by setting the off-diagonal entries of the argument to 0.

Then, we rewrite the objective function in (7.2) as:

\[
f(\beta) = \frac{1}{|T|} \sum_{(i,j,k) \in T} \left[ 1 + \beta^T \nu_{ij} - \beta^T \nu_{ik} \right]_+ + \frac{\lambda}{2} \| \beta \|^2
\]

Let

\[
T^+(\beta) = \{(i,j,k) \in T : 1 + \beta^T \nu_{ij} - \beta^T \nu_{ik} > 0\}
\]

denote the subset of \( T \) that results in strictly positive summands in the objective of (7.2). Then, we
7. Combining Dissimilarities for Nearest Neighbor Classification

can compute the following subgradient of our objective:

$$\nabla_+ f(\beta) = \frac{1}{|T|} \sum_{(i,j,k) \in T^+} (\nu_{ij} - \nu_{ik}) + \lambda \beta.$$ 

Let

$$\beta_* = \arg\min_{\beta} f(\beta)$$

denote the minimizer of $f$. [73, Lemma 4] establishes that

$$\|\beta_*\| \leq 1/\sqrt{\lambda}$$

despite being intended for the minimizer of a soft-margin support vector machine. Let $\Pi$ denote the projection onto the ball $\{\beta : \|\beta\| \leq 1/\sqrt{\lambda}\}$. The projected subgradient method is then described by the iterative update:

$$\beta_i = \Pi (\beta_{i-1} - \eta_i \nabla_+ f(\beta_{i-1}))$$

where $\eta_i$ controls step length. As noted in [74] it is often sufficient to set $\eta_i$ equal to some constant $\eta$. The algorithm terminates when the norm of the computed subgradient gets sufficiently small.

7.4 Numerical Experiments

We performed two numerical experiments using CODI-KNN to learn a dissimilarity function for the $k$-NN classifier. Section 7.4.1 describes an experiment predicting the subject matter in a set of captioned images using a pairwise dissimilarity comparing captions and a pairwise dissimilarity comparing image content. Section 7.4.2 describes an experiment predicting the structure of proteins using twelve different measures of pairwise dissimilarity.
7.4.1 Captioned Images

In this section we describe an experiment classifying captioned images from a Yahoo! database. The database was queried for the keyword “tiger” and 1600 resulting images were classified into one of six classes. In this experiment, we extracted 300 images associated with each of two classes: the golfer Tiger Woods and the Sri Lankan rebels. Dissimilarity matrices comparing image content and text content were computed pairwise for all 600 images. We randomly split the dataset into training and testing sets and replicate the experiment 30 times. We classify the images using the 5-NN classifier constructed using the image dissimilarity alone, the caption dissimilarity alone, the sum of the squared image and caption dissimilarities, and the CODI-KNN dissimilarity. For CODI-KNN, we use the triples:

\[ T = \{(i, j, k) : x_j \in N^W_3(x_i; \delta_{SS}), x_k \in N^B_3(x_i; \delta_{SS}), i = 1, \ldots, N\}. \]

Table 7.1 has the classification accuracy rates for the four techniques over the 30 replications. The results show that the CODI-KNN dissimilarity performs the best, however, it is only moderately better than simply summing the squared dissimilarities.

7.4.2 Protein Structure Prediction

We demonstrate the merits of combining dissimilarities for the k-NN classifier on a problem in protein structure prediction. This dataset has been studied in [28], [75], and [22] and consists of 694 labeled instances, 12 different feature spaces or representations, and 27 classes. The goal is to simultaneously use the various feature representations to predict the structure of the protein
7. Combining Dissimilarities for Nearest Neighbor Classification

<table>
<thead>
<tr>
<th>Replication</th>
<th>Caption</th>
<th>Image</th>
<th>Sum of Squares</th>
<th>CODI-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.635</td>
<td>0.695</td>
<td>0.750</td>
<td>0.760</td>
</tr>
<tr>
<td>2</td>
<td>0.590</td>
<td>0.710</td>
<td>0.745</td>
<td>0.760</td>
</tr>
<tr>
<td>3</td>
<td>0.625</td>
<td>0.760</td>
<td>0.790</td>
<td>0.790</td>
</tr>
<tr>
<td>4</td>
<td>0.625</td>
<td>0.715</td>
<td>0.745</td>
<td>0.715</td>
</tr>
<tr>
<td>5</td>
<td>0.645</td>
<td>0.785</td>
<td>0.795</td>
<td>0.760</td>
</tr>
<tr>
<td>6</td>
<td>0.640</td>
<td>0.700</td>
<td>0.775</td>
<td>0.790</td>
</tr>
<tr>
<td>7</td>
<td>0.635</td>
<td>0.710</td>
<td>0.760</td>
<td>0.760</td>
</tr>
<tr>
<td>8</td>
<td>0.625</td>
<td>0.715</td>
<td>0.720</td>
<td>0.720</td>
</tr>
<tr>
<td>9</td>
<td>0.610</td>
<td>0.775</td>
<td>0.755</td>
<td>0.735</td>
</tr>
<tr>
<td>10</td>
<td>0.650</td>
<td>0.715</td>
<td>0.775</td>
<td>0.760</td>
</tr>
<tr>
<td>11</td>
<td>0.645</td>
<td>0.740</td>
<td>0.800</td>
<td>0.805</td>
</tr>
<tr>
<td>12</td>
<td>0.650</td>
<td>0.730</td>
<td>0.775</td>
<td>0.750</td>
</tr>
<tr>
<td>13</td>
<td>0.610</td>
<td>0.745</td>
<td>0.750</td>
<td>0.785</td>
</tr>
<tr>
<td>14</td>
<td>0.615</td>
<td>0.710</td>
<td>0.755</td>
<td>0.765</td>
</tr>
<tr>
<td>15</td>
<td>0.620</td>
<td>0.710</td>
<td>0.760</td>
<td>0.765</td>
</tr>
<tr>
<td>16</td>
<td>0.625</td>
<td>0.735</td>
<td>0.790</td>
<td>0.795</td>
</tr>
<tr>
<td>17</td>
<td>0.655</td>
<td>0.710</td>
<td>0.755</td>
<td>0.775</td>
</tr>
<tr>
<td>18</td>
<td>0.580</td>
<td>0.725</td>
<td>0.760</td>
<td>0.770</td>
</tr>
<tr>
<td>19</td>
<td>0.645</td>
<td>0.745</td>
<td>0.800</td>
<td>0.800</td>
</tr>
<tr>
<td>20</td>
<td>0.610</td>
<td>0.735</td>
<td>0.790</td>
<td>0.785</td>
</tr>
<tr>
<td>21</td>
<td>0.640</td>
<td>0.735</td>
<td>0.755</td>
<td>0.780</td>
</tr>
<tr>
<td>22</td>
<td>0.610</td>
<td>0.725</td>
<td>0.775</td>
<td>0.785</td>
</tr>
<tr>
<td>23</td>
<td>0.635</td>
<td>0.730</td>
<td>0.735</td>
<td>0.735</td>
</tr>
<tr>
<td>24</td>
<td>0.575</td>
<td>0.700</td>
<td>0.760</td>
<td>0.800</td>
</tr>
<tr>
<td>25</td>
<td>0.645</td>
<td>0.730</td>
<td>0.775</td>
<td>0.780</td>
</tr>
<tr>
<td>26</td>
<td>0.620</td>
<td>0.715</td>
<td>0.755</td>
<td>0.785</td>
</tr>
<tr>
<td>27</td>
<td>0.595</td>
<td>0.770</td>
<td>0.770</td>
<td>0.775</td>
</tr>
<tr>
<td>28</td>
<td>0.640</td>
<td>0.705</td>
<td>0.755</td>
<td>0.755</td>
</tr>
<tr>
<td>29</td>
<td>0.655</td>
<td>0.695</td>
<td>0.755</td>
<td>0.775</td>
</tr>
<tr>
<td>30</td>
<td>0.645</td>
<td>0.780</td>
<td>0.740</td>
<td>0.755</td>
</tr>
</tbody>
</table>

| Mean        | 0.627   | 0.728 | 0.764          | 0.769    |
| St. Dev     | 0.022   | 0.025 | 0.020          | 0.023    |

Table 7.1: Classification accuracy rates of the 5-NN classifier.
which is encoded into one of the 27 classes. The results presented in this section are not competitive with the best results in the above papers, however, they do demonstrate that the $k$-NN classifier can benefit from combining dissimilarities using CODI-KNN as compared to simply adding squared dissimilarities or using the best individual dissimilarity. The dataset was obtained from http://mkl.ucsd.edu/ and consists of blocks of features labeled: (1) Composition, (2) Secondary, (3) Hydrophobicity, (4) Volume, (5) Polarity, (6) Polarizability, (7) L1, (8) L4, (9) L14, (10) L30, (11) SWblosum62, and (12) SWpam50 (see the website and associated papers for additional details). For the first ten dissimilarity matrices we compute a second order heterogeneous polynomial kernel using each of the first ten feature representations. Then, using these inner product matrices we compute the respective Euclidean distances via the transform: $\delta^2_{ij} = K_{ii} + K_{jj} - K_{ij} - K_{ji}$. For the last two representations we take the standard Euclidean distance.

Unlike the experiments in the papers cited above we randomize the training and testing sets and compute accuracy rates (percentage of proteins in the testing set correctly labeled by the classifier) over 20 replications. For each replication we assign each instance to the training and testing sets such that for each class there are even number in each. When an odd number of proteins are in a class we put one extra protein in the training set. We used the 5-NN classifier for all of the following experiments. First, we computed the accuracy rates using each of the twelve dissimilarities independently (see Table 7.2).

Next, we compare the performance of the 5-NN classifier using $\delta_{SS}$ (the sum of the squared dissimilarities) and $\delta_*$ (the dissimilarity computed using CODI-KNN). For CODI-KNN we used the triples:

$$T = \{(i, j, k) : x_j \in N_3^W(x_i; \delta_{SS}), x_k \in N_3^B(x_i; \delta_{SS}), i = 1, \ldots, N\}.$$ 

We also investigated using small subsets of the input dissimilarity matrices for both methods. We
7. Combining Dissimilarities for Nearest Neighbor Classification

<table>
<thead>
<tr>
<th>Feature</th>
<th>Accuracy (1 St. Dev.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composition</td>
<td>0.312 (0.015)</td>
</tr>
<tr>
<td>Secondary</td>
<td>0.331 (0.019)</td>
</tr>
<tr>
<td>Hydrophobicity</td>
<td>0.186 (0.017)</td>
</tr>
<tr>
<td>Volume</td>
<td>0.188 (0.014)</td>
</tr>
<tr>
<td>Polarity</td>
<td>0.178 (0.019)</td>
</tr>
<tr>
<td>Polarizability</td>
<td>0.173 (0.018)</td>
</tr>
<tr>
<td>L1</td>
<td>0.275 (0.016)</td>
</tr>
<tr>
<td>L4</td>
<td>0.236 (0.017)</td>
</tr>
<tr>
<td>L14</td>
<td>0.190 (0.016)</td>
</tr>
<tr>
<td>L30</td>
<td>0.148 (0.019)</td>
</tr>
<tr>
<td>SWblosum62</td>
<td>0.360 (0.018)</td>
</tr>
<tr>
<td>SWpam50</td>
<td>0.340 (0.016)</td>
</tr>
</tbody>
</table>

Table 7.2: Classification accuracy rates of the 5-NN classifier for individual dissimilarity matrices.

<table>
<thead>
<tr>
<th># of $\Delta$s</th>
<th>Sum of Squares (1 St. Dev.)</th>
<th>CODI-KNN (1 St. Dev.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.368 (0.032)</td>
<td>0.368 (0.033)</td>
</tr>
<tr>
<td>3</td>
<td>0.333 (0.026)</td>
<td>0.338 (0.028)</td>
</tr>
<tr>
<td>4</td>
<td>0.361 (0.017)</td>
<td>0.379 (0.020)</td>
</tr>
<tr>
<td>5</td>
<td>0.348 (0.019)</td>
<td>0.370 (0.022)</td>
</tr>
<tr>
<td>6</td>
<td>0.350 (0.017)</td>
<td>0.362 (0.022)</td>
</tr>
<tr>
<td>7</td>
<td>0.334 (0.021)</td>
<td>0.357 (0.019)</td>
</tr>
<tr>
<td>8</td>
<td>0.317 (0.017)</td>
<td>0.349 (0.024)</td>
</tr>
<tr>
<td>9</td>
<td>0.310 (0.017)</td>
<td>0.343 (0.023)</td>
</tr>
<tr>
<td>10</td>
<td>0.296 (0.018)</td>
<td>0.342 (0.021)</td>
</tr>
<tr>
<td>11</td>
<td>0.294 (0.015)</td>
<td>0.340 (0.019)</td>
</tr>
<tr>
<td>12</td>
<td>0.291 (0.014)</td>
<td>0.340 (0.020)</td>
</tr>
</tbody>
</table>

Table 7.3: Classification accuracy rates of $k$-NN classifier for combined dissimilarity matrices.

ranked the twelve dissimilarity matrices according to their 5-NN leave-one-out classification accuracy rates within the training set. Then, we chose subsets using the best two matrices, best three matrices, etc., for use by each approach. The results are in Table 7.3.
7. Combining Dissimilarities for Nearest Neighbor Classification

Discussion

Using multiple disparate feature representations poses a difficult challenge in pattern recognition. When multiple dissimilarities are available one approach is to use them to learn a new dissimilarity for use by the $k$-NN classifier. We propose a method for learning the coefficients of a heterogeneous polynomial of degree two in the original dissimilarities to improve the performance of the $k$-NN classifier. Numerical experiments show that adding the sum of the squared dissimilarities can perform better than simply using individual dissimilarities. This is akin to the now conventional wisdom in the MKL community that adding kernels often performs as well as or better than learning convex combinations. In both experiments, however, we show that even though the CODI-KNN + $k$-NN approach is not state of the art, it performs better than simply adding the squared dissimilarities.

The CODI-KNN approach as described in this report hinges upon an initial selection of the within-class and between-class nearest neighbors. Doing so implies that one has knowledge of a meaningful measure of dissimilarity a priori, however, the purpose of CODI-KNN is to learn a more useful measure of dissimilarity. To address this disconnect one could consider iterating CODI-KNN such that the within-class and between-class nearest neighbor graphs are constructing using dissimilarities learned from a previous iteration.
Matched Pair Testing

8.1 Introduction

Data mining often necessitates the fusion of disparate sources of data. For example, a captioned image has both text and an image that describe the same object, and images may be compared by measures of color, texture, etc. Often, domain experts have constructed specialized measures of proximity for each data source, but the measures are not on commensurate scales. Thus, a fundamental problem in data fusion is the problem of constructing a suitable representation of the data in which inference can proceed.

In machine learning, recent approaches to combining disparate data have assumed the existence of pairwise similarities, interpreted as inner products (kernels). From a set of base kernels, multiple kernel learning (MKL) constructs a single kernel, typically for use in a support vector machine. See Lanckriet et al. [49] and Sonnenburg et al. [76] for key references and McFee and Lanckriet [57] for a rare example of a nonmetric approach. MKL is often used for multi-view learning, i.e., learning about objects that are observed from multiple views. MKL was used in Varma and Ray [90] to combine information from multiple base features for the purpose of classifying images. A Bayesian
approach to MKL for multi-view learning was proposed in Christoudias et al. [14]. MKL has also been used to combine kernels for use across multiple tasks [38].

We consider the problem of matched pair hypothesis testing, i.e., the problem of determining whether or not a pair of disparate feature vectors correspond to a common object. For example, we may be interested in determining whether two photographs taken under two different conditions (e.g., a passport photo and a security camera still) contain the same face. Instead of working with pairwise similarities and positive semidefinite kernels, we work with pairwise dissimilarities and Euclidean distances. Instead of learning a convex combination of multiple kernels for the purpose of classification, we embed multiple dissimilarities in a common Euclidean representation for the purpose of hypothesis testing. Our emphasis on dissimilarities resembles Pękalska and Duin [64]. Methods for constructing configurations of points from pairwise dissimilarities is the province of multidimensional scaling (MDS). See Borg and Groenen [5] for an introduction to MDS.

Priebe et al. [67] proposed an embedding approach for matched pair hypothesis testing. Given two sets of pairwise dissimilarities for a training set of matched pairs, their JOFC approach constructs two distinct configurations in the same Euclidean space in such a way that elements of a matched pair tend to be near each other. A test pair is then embedded, each element with respect to the corresponding configuration. The Euclidean distance between the locations of the embedded test pair is then used as a test statistic. We propose an alternative approach that uses three-way MDS to construct a single configuration of points from two sets of pairwise dissimilarities. To facilitate the fusion of disparate measures of dissimilarity, our approach is nonmetric, i.e., only the ranks of the pairwise dissimilarities are used to construct the common representation of the matched pairs and to embed the elements of test pairs.

Section 8.2 introduces notation and describes the JOFC methodology in [67]. We modify JOFC in Section 8.3, deriving a three-way nonmetric MDS approach to matched pair hypothesis testing.
8. Matched Pair Testing

Section 8.4 summarizes the algorithms that implement our approach. Section 8.5 provides an illustrative example. A reader unfamiliar with embedding methods might benefit from examining the figures in this section before studying the details of the formulations. Section 8.6 reports results from several numerical experiments and Section 8.7 concludes.

8.2 Background

Let \( \omega_1, \omega_2, \ldots, \omega_N \in \Omega \) be independent and identically distributed objects drawn from an abstract probability space. The objects may or may not be directly observable, but the feature maps \( \pi_k : \Omega \mapsto \Xi_k \) represent the objects in observable feature spaces. Feature spaces are often Euclidean, but we allow more general representations, e.g., graphs.

We restrict attention to the case of \( K = 2 \) feature maps. If \( x_{i1} = \pi_1(\omega_i) \in \Xi_1 \) and \( x_{i2} = \pi_2(\omega_i) \in \Xi_2 \) for some \( \omega_i \in \Omega \), then we say that \((x_{i1}, x_{i2})\) is a matched pair and we write \( x_{i1} \sim x_{i2} \). If \( x_{i1} = \pi_1(\omega_i) \) and \( x_{j2} = \pi_2(\omega_j) \) for some \( \omega_i \neq \omega_j \), then we say that \((x_{i1}, x_{j2})\) is a mismatched pair and we write \( x_{i1} \not\sim x_{j2} \). Unless the feature maps are injective, a pair may be both matched and mismatched.

We assume that each feature space is equipped with a measure of dissimilarity between pairs of points. Formally, a measure of dissimilarity is a function \( \delta_k : \Xi_k \times \Xi_k \mapsto \mathbb{R} \) that is symmetric \( (\delta_k(x_{ik}, x_{jk}) = \delta_k(x_{jk}, x_{ik})) \), nonnegative, and zero if its arguments are identical. Informally, the dissimilarity of \( \pi_k(\omega_i) \) and \( \pi_k(\omega_j) \) should be small if \( \omega_i \) and \( \omega_j \) are similar, large if they are dissimilar. In many applications, domain experts have developed specialized measures of dissimilarity, e.g., for comparing the color or texture of images. Absent such measures, one often relies on standard measures of distance, e.g., Euclidean distance between feature vectors or shortest path distance between nodes of a graph.
8. Matched Pair Testing

8.2.1 Matched Pair Hypothesis Testing

Consider two probability models for generating pairs in $\Xi_1 \times \Xi_2$: either ($H_0$) draw a single object, $\omega_i$, then compute $\pi_1(\omega_i)$ and $\pi_2(\omega_i)$; or ($H_A$) draw two objects, $\omega_i$ and $\omega_j$, independently, then compute $\pi_1(\omega_i)$ and $\pi_2(\omega_i)$. We assume that a training set of matched pairs,

$$S_N = \{(x_{11}, x_{12}), \ldots, (x_{N1}, x_{N2})\},$$

was generated by $H_0$. If it is not known whether $(y_1, y_2)$ was generated by $H_0$ or $H_A$, then $(y_1, y_2)$ is a test pair. Given a training set and a test pair, we propose a test of the null hypothesis that the test pair was generated by $H_0$ against the alternative hypothesis that the test pair was generated by $H_A$. We write these simple hypotheses as $H_0 : y_1 \sim y_2$ and $H_A : y_1 \not\sim y_2$. This formulation of matched pair hypothesis testing is a special case (pairs instead of $K$-tuples) of a more general decision problem described in [67].

8.2.2 Joint Optimization of Fidelity and Commensurability (JOFC)

[67] described an approach to matched pair hypothesis testing that they termed Omnibus Embedding. For each dissimilarity function $\delta_k$, they used MDS to represent the training objects in a $p$-dimensional Euclidean space. The two representations were aligned in a manner described below. Each $y_k$ was then embedded in representation $k$ and the Euclidean distance between the embedded points was used as a test statistic.

More precisely, let

$$\Delta_k = [\delta_k(x_{ik}, x_{jk})]_{ij}$$

and let $\tilde{x}_{1k}, \tilde{x}_{2k}, \ldots, \tilde{x}_{Nk} \in \mathbb{R}^p$ denote the corresponding configuration of points constructed by
minimizing the raw stress criterion,

\[
\sigma_r(\bar{X}_k; \Delta_k) = \sum_{i<j} \left[ \delta_k (x_{ik}, x_{jk}) - d(\bar{x}_{ik}, \bar{x}_{jk}) \right]^2
\]

\[
= \frac{1}{2} \| \Delta_k - D(\bar{X}_k) \|_F^2,
\]

where \( d \) denotes Euclidean distance in \( \mathbb{R}^p \), \( \bar{X}_k \) is the \( N \times p \) configuration matrix that contains \( \bar{x}_i^T \) in row \( i \), \( D(\bar{X}_k) = [d(\bar{x}_{ik}, \bar{x}_{jk})]_{ij} \) is the corresponding matrix of pairwise Euclidean distances, and \( \| \cdot \|_F \) denotes the Frobenius norm. [67] termed \( \sigma_r(\bar{X}_k; \Delta_k) \) the fidelity of embedding \( \Delta_k \). Aligning \( \bar{X}_1 \) and \( \bar{X}_2 \) requires minimizing a measure of commensurability, e.g.,

\[
\eta(\bar{X}_1, \bar{X}_2) = \| \bar{X}_1 - \bar{X}_2 \|_F^2.
\]

One might simply compose a Procrustes analysis with MDS \((p \circ m)\), first embedding to minimize fidelity, then rotating one configuration to improve commensurability. In contrast, [67] proposed jointly optimizing fidelity and commensurability \((JOFC)\). Their general framework for JOFC allows one to choose various measures of commensurability and to vary the tradeoff between fidelity and commensurability. One of the key insights in [67] is that jointly optimizing fidelity and commensurability leads to a more powerful test than does optimizing each in turn, e.g., by \( p \circ m \).

We present a variation of JOFC that minimizes

\[
\sigma_j(\bar{X}_1, \bar{X}_2) = \sigma_r(\bar{X}_1; \Delta_1) + \sigma_r(\bar{X}_2; \Delta_2) + \eta(\bar{X}_1, \bar{X}_2)
\]

\[
= \frac{1}{2} \sum_{k=1,2} \| \Delta_k - D(\bar{X}_k) \|_F^2 + \| \bar{X}_1 - \bar{X}_2 \|_F^2.
\]
subject to $\tilde{X}_1, \tilde{X}_2 \in \mathbb{R}^{N \times p}$, $p$ fixed. Writing $\tilde{X}_1 = \tilde{X}_1 Q$ for an orthogonal matrix $Q$, we see that $p \circ m$ first minimizes $\sigma_r(\tilde{X}_1; \Delta_1) + \sigma_r(\tilde{X}_2; \Delta_2)$, then minimizes $\|\tilde{X}_1 - \tilde{X}_2\|_F^2 = \|\tilde{X}_1 Q - \tilde{X}_2\|_F^2$ with respect to $Q$. In general, $p \circ m$ does not minimize $\sigma_j$. Figure 8.1 displays an embedding of 20 matched pairs by JOFC. One configuration is represented by circles, the other by crosses, and the pairs are connected by line segments.

Now suppose that $(y_1, y_2)$ have dissimilarity vectors

$$\bar{\delta}_k = (\delta_k (y_{1k}, x_{1k}), \ldots, \delta_k (y_{Nk}, x_{Nk}))^T.$$  

Each $y_k$ is embedded in relation to $\bar{X}_k$ by minimizing

$$\sigma_{j-o}(\bar{y}_k) = \frac{1}{2} \left\| \begin{bmatrix} \Delta_k & \bar{\delta}_k \\ \bar{\delta}_k^T & 0 \end{bmatrix} - D \left( \begin{bmatrix} \bar{X}_k \\ \bar{y}_k^T \end{bmatrix} \right) \right\|_F^2,$$  

Figure 8.1: JOFC embedding of 20 matched pairs.
the raw stress criterion for an augmented dissimilarity matrix and configuration in which only one point is free to vary. This is an example of out-of-sample embedding. Figure 8.2 illustrates the out-of-sample embedding of three matched pairs, whereas Figure 8.3 illustrates the out-of-sample embedding of three mismatched pairs. In both cases, the reference configuration is the configuration displayed in Figure 8.1.

Figure 8.2: Three matched pairs have been added to the configuration in Figure 8.1.

If both fidelity and commensurability are good, then the Euclidean distance $d(\tilde{y}_1, \tilde{y}_2)$ should be small for matched pairs and larger for mismatched pairs. This distance is the JOFC test statistic. Its null distribution can be estimated from the training set by cross-validation, then used to determine critical values for achievable levels of significance. For a given critical value, power can be estimated from the empirical distribution of $d(\tilde{y}_1, \tilde{y}_2)$ for mismatched pairs.
8. Matched Pair Testing

Figure 8.3: Three mismatched pairs have been added to the configuration in Figure 8.1.

8.3 Three-Way Nonmetric MDS (3WNM)

Inspired by a comment at the end of [67], that it might be of interest to compare three-way MDS to JOFC, we now describe an alternative approach to matched pair hypothesis testing. In contrast to JOFC, three-way MDS constructs a single configuration from both $\Delta_k$. Because the $\delta_k$ may measure dissimilarity in different ways, e.g., Euclidean distance versus shortest path distance, we only use information about how the entries of each $\Delta_k$ are ordered, i.e., we allow monotonic transformations of each $\delta_k$. Thus, our approach to embedding is an example of three-way nonmetric MDS. For each test pair embedded as $\tilde{y}_1, \tilde{y}_2 \in \mathbb{R}^p$, we use $d(\tilde{y}_1, \tilde{y}_2)$ as a test statistic, as described in Section 8.2.2.

We describe 3WNM for the special case of matched pairs ($K = 2$), but note that our formulation naturally extends to $K > 2$ representations. The test statistic is then the square root of the sum of the squared distances between the $K$ out-of-sample embedded points.
8. Matched Pair Testing

8.3.1 Embedding the Training Pairs

Fix $p$, the embedding dimension, and let $\tilde{X}$ denote the $N \times p$ configuration matrix constructed from the training pairs. In three-way MDS, the common representation constructed from multiple dissimilarity matrices is often called the group space. Because $\tilde{X}$ models both $\Delta_1$ and $\Delta_2$, a natural extension of the raw stress criterion is

$$
\sigma_r \left( \tilde{X}; \Delta_1 \right) + \sigma_r \left( \tilde{X}; \Delta_2 \right) = \frac{1}{2} \sum_{k=1,2} \left\| \Delta_k - D \left( \tilde{X} \right) \right\|_F^2,
$$

the identity model for three-way MDS [15]. More generally, one might weight the summands. Notice that the identity model is just (8.1) with $\tilde{X}_1 = \tilde{X}_2$.

Nonmetric MDS constructs configurations using only the ranks of the dissimilarities. Following [45], we replace each $\Delta_k$ with $\mathcal{M}(\Delta_k)$, the cone of all dissimilarity matrices whose entries have the same order as the entries of $\Delta_k$, i.e.,

$$
\mathcal{M}(\Delta_k) = \left\{ \tilde{\Delta}_k : \begin{array}{l}
\delta_k(x_{ik}, x_{jk}) \leq \bar{\delta}_k(x_{mk}, x_{nk}) \quad \text{if} \\
\delta_k(x_{ik}, x_{jk}) \leq \delta_k(x_{mk}, x_{nk})
\end{array} \right\}
$$

Equivalently, $\mathcal{M}(\Delta_k)$ is the cone of all dissimilarity matrices that can be obtained from $\Delta_k$ by a nondecreasing transformation of its entries. Note that $0 \in \mathcal{M}(\Delta_k)$. We will refer to the elements of $\mathcal{M}(\Delta_k)$ as surrogate dissimilarity matrices.

Conceptually, we would like to find a triple $(\tilde{X}, \tilde{\Delta}_1, \tilde{\Delta}_2)$ that minimizes

$$
\sigma_n \left( \tilde{X}, \tilde{\Delta}_1, \tilde{\Delta}_2 \right) = \frac{1}{2} \sum_{k=1,2} \left\| \tilde{\Delta}_k - D \left( \tilde{X} \right) \right\|_F^2
$$

subject to $\tilde{\Delta}_k \in \mathcal{M}(\Delta_k)$. Unfortunately, embedding each point at the same location (resulting
8. Matched Pair Testing

in \( D(\tilde{X}) = 0 \) and choosing \( \Delta_k = 0 \) results in a degenerate global solution. To preclude such solutions, we are obliged to modify the problem. Traditionally, degenerate solutions have been precluded by adopting a scale-invariant objective function. Instead, following the observation in [84] that degenerate solutions can be precluded by imposing explicit constraints, we require the surrogate dissimilarity matrices to lie in

\[
\mathcal{N} = \left\{ \tilde{\Delta}_k : \| \tilde{\Delta}_k \|_F^2 \geq 1 \right\}.
\]

Our optimization problem is then

\[
\begin{align*}
\text{minimize} & \quad \sigma_n(\tilde{X}, \tilde{\Delta}_1, \tilde{\Delta}_2) \\
\text{subject to} & \quad \tilde{\Delta}_k \in \mathcal{M}(\Delta_k) \cap \mathcal{N}.
\end{align*}
\]

(8.2)

Thus, 3WNM constructs a single \( \tilde{X} \) that attempts to respect the orderings of both \( \Delta_1 \) and \( \Delta_2 \).

8.3.2 Embedding the Test Pairs

After embedding the training pairs in \( \mathbb{R}^p \), we embed the test pairs in relation to the training pairs. Given a pair of test objects \((y_1, y_2)\), let

\[
\bar{\delta}_k = (\delta_k(y_k, x_{1k}), \ldots, \delta_k(y_k, x_{Nk}))^T \text{ for } k = 1, 2
\]

denote the vector of \( \delta_k \)-dissimilarities between \( y_k \) and each of the objects in the training set. Our out-of-sample embedding of \( y_1 \) and \( y_2 \) resembles the technique described in Section 8.2.2; however, we embed with respect to a single configuration and allow monotonic transformations of \( \bar{\delta}_k \).
Hence, we obtain $\tilde{y}_1, \tilde{y}_2 \in \mathbb{R}^p$ by minimizing

$$\sigma_{n-o}(\tilde{y}_k, \tilde{\delta}_k) = \frac{1}{2} \left\| \begin{bmatrix} \tilde{\Delta}_k & \tilde{\delta}_k \\ \tilde{\delta}_k^T & 0 \end{bmatrix} - D \left( \begin{bmatrix} \tilde{X} \\ \tilde{\delta}_k^T \tilde{y}_k \end{bmatrix} \right) \right\|_F^2,$$

subject to the constraint that

$$\begin{bmatrix} \tilde{\Delta}_k & \tilde{\delta}_k \\ \tilde{\delta}_k^T & 0 \end{bmatrix} \in \mathcal{M} \left( \begin{bmatrix} \Delta_k & \delta_k \\ \delta_k^T & 0 \end{bmatrix} \right).$$

Because $\tilde{\Delta}_k$ is fixed, it is not necessary to impose an additional nondegeneracy constraint. Notice that (8.3) reduces to simple bound constraints on the components of $\tilde{\delta}_k$, the bounds determined by entries in $\tilde{\Delta}_k$.

### 8.4 Algorithms

We now describe algorithms for finding good solutions of (8.2) and its out-of-sample extension. Our algorithms rely on a general search strategy, sometimes called variable alternation, in which the objective function is successively minimized with respect to distinct blocks of decision variables. The classic example of minimization by variable alternation is coordinate descent, in which the blocks consist of the individual decision variables. Variable alternation produces a non-decreasing sequence of objective function values. Under mild conditions, the sequence of iterates converges to a connected set of stationary points.

The optimization problems formulated in Section 8.3 have a natural block structure. For (8.2), the blocks are $\tilde{X}, \tilde{\Delta}_1,$ and $\tilde{\Delta}_2$. For the out-of-sample problem, the blocks are $\tilde{y}_k$ and $\tilde{\delta}_k$. 
8. Matched Pair Testing

8.4.1 Three-Way Nonmetric MDS (3WNM)

Algorithm 8.1 describes our variable alternation strategy for solving (8.2). Many MDS algorithms have this general structure [5]. The alternating subproblems are solved by Algorithms 8.2 and 8.3. The loop may be terminated by imposing a maximum number of iterations ($\ell \leq L$) and/or by monitoring how much an iteration decreases the value of the objective function. If $N$ is large, then one might choose $L$ small. In practice, much of the progress is made in the first several iterations.

Algorithm 8.1. 3WNM

1. Compute $\sigma_n^{[0]}$ and set $\ell = 0$.

2. Do until termination:

   (a) Increment $\ell$.

   (b) Fix $\tilde{\Delta}_1$ and $\tilde{\Delta}_2$ and minimize $\sigma_n$ with respect to $\tilde{X}$. This step uses Algorithm 8.2.

   (c) Fix $\tilde{X}$ and minimize $\sigma_n$ with respect to $\tilde{\Delta}_1$ and $\tilde{\Delta}_2$, subject to $\tilde{\Delta}_k \in \mathcal{M}(\Delta_k) \cap \mathcal{N}$. This step uses Algorithm 8.3.

   (d) Compute $\sigma_n^{[\ell]}$.

Step 2b in Algorithm 8.1 fixes the surrogate dissimilarity matrices $\tilde{\Delta}_1$ and $\tilde{\Delta}_2$ and minimizes $\sigma_n$ with respect to $\tilde{X}$. This problem is the standard identity model for three-way MDS [15]. [5, pages 129]
8. Matched Pair Testing

[476–477] simplified this problem by writing

$$\sigma_n(\tilde{X}; \tilde{\Delta}_1, \tilde{\Delta}_2) = \frac{1}{2} \sum_{k=1,2} \| \Delta_k - D(\tilde{X}) \|_F^2$$

$$= \| \Delta - D(\tilde{X}) \|_F^2 + \frac{1}{2} \| \Delta - \tilde{\Delta}_1 \|_F^2 + \frac{1}{2} \| \Delta - \tilde{\Delta}_2 \|_F^2,$$

where $\Delta = (\tilde{\Delta}_1 + \tilde{\Delta}_2)/2$. Only the first term in the final expression depends on $\tilde{X}$, and this term is just the raw stress criterion with the average surrogate dissimilarities. Hence, Step 2b requires only an algorithm that minimizes the raw stress criterion for a single fixed dissimilarity matrix.

The raw stress criterion is often minimized by iterative majorization, specifically by repeated application of a fixed point mapping known as the Guttman transform. If, as here, the summands in the raw stress criterion are equally weighted, then computing the Guttman transform is not expensive. Let $\delta_{ij}$ denote entry $ij$ in $\Delta$ and let $d_{ij}(\tilde{X})$ denote $\| \tilde{x}_i - \tilde{x}_j \|$. Let $B(\tilde{X})$ denote the $N \times N$ matrix with off-diagonal entries $b_{ij} = -\delta_{ij}/d_{ij}(\tilde{X})$ and diagonal entries $b_{ii} = -\sum_{j: i \neq j} b_{ij}$. Then the Guttman transform of $\tilde{X}$ is

$$\Gamma(\tilde{X}) = B(\tilde{X})\tilde{X}/N.$$

See [5, pages 187–191] for the derivation of $\Gamma$.

The complete algorithm for step 2b is described in Algorithm 8.2. Again, the loop may be terminated by imposing a maximum number of iterations and/or by monitoring how much an iteration decreases the value of the raw stress criterion. In our experience, 5–10 iterations usually produce a good $\tilde{X}$.

**Algorithm 8.2. Update $\tilde{X}$**

1. Compute $\Delta = (\tilde{\Delta}_1 + \tilde{\Delta}_2)/2$.
2. Compute $\sigma_n^{[0]}$ and set $\ell' = 0$. 

130
3. Do until termination:

(a) Increment $\ell'$.

(b) $\tilde{X} \leftarrow \Gamma(\tilde{X})$

(c) Compute $\sigma_n^{[\ell']}$. 

Step 2c in Algorithm 8.1 updates the surrogate dissimilarity matrices. For a fixed $\tilde{X}$, the objective function $\sigma_n$ is separable and it suffices to update each $\tilde{\Delta}_k$ by projecting $D(\tilde{X})$ into $M(\Delta_k) \cap N$. This projection is accomplished by first projecting $D(\tilde{X})$ into $M(\Delta_k)$, then (if necessary) rescaling the solution. Projection into order constraints is called isotonic regression, for which various algorithms exist. For the case of a complete ordering, [36] describe an algorithm that is linear in the number of variables, hence $O(N^2)$ in our application. Algorithms for general partial orderings are quadratic in the number of variables, hence $O(N^4)$ in our application. If $M(\Delta_k)$ only defines a partial ordering (because there are ties in the entries of $\Delta_k$), then we circumvent the expense of partial ordering by imposing a complete ordering. If $\delta_{ij} = \delta_{k\ell}$ and $d_{ij}(\tilde{X}) < d_{k\ell}(\tilde{X})$, then we impose $\tilde{\delta}_{ij} \leq \tilde{\delta}_{k\ell}$. In the unlikely event that $\delta_{ij} = \delta_{k\ell}$ and $d_{ij}(\tilde{X}) = d_{k\ell}(\tilde{X})$, then we randomly impose either $\tilde{\delta}_{ij} \leq \tilde{\delta}_{k\ell}$ or $\tilde{\delta}_{ij} \geq \tilde{\delta}_{k\ell}$. Admittedly, our approach is ad hoc. If we impose additional order constraints, we may not actually compute the projection of $D(\tilde{X})$ into $M(\Delta_k)$. Furthermore, how we modify $M(\Delta_k)$ may vary from iteration to iteration of Algorithm 8.1. Nevertheless, our approach appears to work well in practice.

**Algorithm 8.3. Update $\tilde{\Delta}_k$**

1. Compute $D(\tilde{X})$.

2. Update $\tilde{\Delta}_k$ by projecting $D(\tilde{X})$ into $M(\Delta_k)$, suitably modified to define a complete ordering, using the algorithm described in [36].
3. If $\|\tilde{\Delta}_k\|_F^2 < 1$, then set $\tilde{\Delta}_k = \tilde{\Delta}_k / \|\tilde{\Delta}_k\|$.

The primary computational burden of 3WNN lies in updating $\tilde{\Delta}_1$ and $\tilde{\Delta}_2$. Note that these updates can be performed in parallel. Furthermore, as described in [40], isotonic regression can itself be parallelized efficiently.

### 8.4.2 Out-of-sample Embedding

Trosset and Priebe [89] proposed an algorithm for out-of-sample embedding with respect to the error criterion used in classical MDS. They also observed that out-of-sample embedding has a natural formulation with respect to the raw stress criterion. Ma and Priebe [56] derived an iterative majorization algorithm for this case. Here we describe an algorithm for out-of-sample embedding in which the raw stress criterion is minimized subject to order constraints on a set of surrogate dissimilarities.

The algorithm for embedding a test pair resembles the algorithm for embedding the set of training pairs. Again we use variable alternation and terminate the loop by imposing a maximum number of iterations and/or by monitoring how much an iteration decreases the value of the objective function. The following algorithm is repeated for $k = 1, 2$. Because the greatest expense lies in computing the distances between $\tilde{y}_k$ and each $\tilde{x}_{ik}$, computation is $O(N)$.

**Algorithm 8.4. Out-of-sample Embedding**

0. Compute $\bar{\delta}_k = (\delta_k(y_k, x_{1k}), \ldots, \delta_k(y_k, x_{Nk}))^T$.

1. Initialize $\bar{y}_k$, e.g., by placing $\bar{y}_k$ at the origin or at the $\bar{x}_{ik}$ for which $\bar{\delta}_k(y_k, x_{ik})$ is smallest.

2. Compute $c_{n-o}^{[0]}$ and set $t = 0$.

3. Do until termination:
8. Matched Pair Testing

(a) Increment \( \ell \).

(b) Fix \( \tilde{\delta}_k \) and minimize \( \sigma_{n-o} \) with respect to \( \tilde{y}_k \). This step uses Algorithm 8.5.

(c) Fix \( \tilde{y}_k \) and minimize \( \sigma_{n-o} \) with respect to \( \tilde{\delta}_k \). This step uses Algorithm 8.6.

(d) Compute \( \sigma^{[\ell']}_{n-o} \).

Algorithm 8.5 is a variant of Algorithm 8.2. Instead of varying every point in the configuration, \( N \) points are fixed and one point is varied. Let \( b(\tilde{y}_k) \) have entries \( \tilde{\delta}_{ki}/d(\tilde{y}_k, \tilde{x}_{ik}) \), where \( \tilde{\delta}_{ki} \) is entry \( i \) in \( \tilde{\delta}_k \), and let

\[
\gamma(\tilde{y}_k) = \frac{1}{N} \sum_{i=1}^{N} [(1 - b_i) \tilde{x}_i + b_i \tilde{y}_k].
\]

Then out-of-sample embedding is performed as follows:

**Algorithm 8.5. Update \( \tilde{y}_k \)**

1. Compute \( \sigma_{n-o}^{[0]} \) and set \( \ell' = 0 \).

2. Do until termination:

   (a) Increment \( \ell' \).

   (b) Compute \( d(\tilde{y}_k, \tilde{x}_1), \ldots, d(\tilde{y}_k, \tilde{x}_N) \).

   (c) \( \tilde{y}_k \leftarrow \gamma(\tilde{y}_k) \)

   (d) Compute \( \sigma_{n-o}^{[\ell']} \).

Algorithm 8.6 is a variant of Algorithm 8.3:

**Algorithm 8.6. Update \( \tilde{\delta}_k \)**

1. Compute \( d(\tilde{X}) = (d(\tilde{y}_k, \tilde{x}_1), \ldots, d(\tilde{y}_k, \tilde{x}_N))^T \).

2. Project \( d(\tilde{X}) \) into the rectangle defined by (8.3) and denote the projection \( \tilde{\delta}_k \).
8. Matched Pair Testing

8.5 Illustrative Example

We illustrate 3WNM with a small version of an experiment performed in [67]. The original $N = 20$ objects are drawn from a multivariate normal distribution on $\Omega = \mathbb{R}^3$.

\[
\omega_1, \omega_2, \ldots, \omega_N \sim \text{Normal}(0, I_3).
\]

The feature spaces are $\Xi_1 = \Xi_2 = \mathbb{R}^6$ and the observed feature vectors are

\[
x_{ik} = \pi_k(\omega_i) = ((1 - a)s_{ik}^t, a \epsilon_{ik}^t)^t,
\]

where

\[
s_{ik} \sim \text{Normal}(\omega_i, I_3/30) \quad \text{and} \quad \epsilon_{ik} \sim \text{Normal}(0, I_3).
\]

The $s_{ik}$ represent observations of the original $\omega_{ik}$ that have been corrupted by measurement error, while the $\epsilon_{ik}$ represent pure noise in extraneous dimensions. The constant $a$ controls the relative magnitudes of the $s_{ik}$ and the $\epsilon_{ik}$, hence the extent to which the measured dissimilarities in $\mathbb{R}^3$ will be affected by noise in the extraneous dimensions. We used $a = 0.4$, resulting in a fairly challenging inferential task.

The dissimilarity measure $\delta_k$ was Euclidean distance in $\Xi_k$. We formed $\Delta_1$ and $\Delta_2$, then used 3WNM to embed the $(x_{i1}, x_{i2})$ in $\mathbb{R}^p$. The problem of choosing $p$ is a problem of model selection; we chose $p = 2$ to facilitate visualization. The resulting points, $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{20}$, are displayed in Figure 8.4.

Figures 8.5 and 8.6 each display the $\tilde{x}_i$ and three additional pairs (identified by plotting symbols
8. Matched Pair Testing

Figure 8.4: Illustrative embedding of $N = 20$ matched pairs in $\mathbb{R}^2$ by three-way nonmetric MDS.

1, 2, and 3) embedded out-of-sample. These pairs are matched in the case of Figure 8.5, mismatched in the case of Figure 8.6. The intrapair distances of the matched pairs are much smaller than the intrapair distances of the mismatched pairs. This observation is the basis for our approach to matched pair hypothesis testing: we reject $H_0 : y_1 \sim y_2$ if and only if $d(\tilde{y}_1, \tilde{y}_2)$ is sufficiently large.

We generated 1000 matched and 1000 mismatched $(y_1, y_2)$ and computed $d(\tilde{y}_1, \tilde{y}_2) = \|\tilde{y}_1 - \tilde{y}_2\|$ for each pair. Figure 8.7 displays the empirical cumulative distribution functions (CDFs) of $d(\tilde{y}_1, \tilde{y}_2)$ and Figure 8.8 displays corresponding kernel density estimates. Evidently, $d(\tilde{y}_1, \tilde{y}_2)$ is stochastically larger under the alternative hypothesis of mismatched pairs than under the null hypothesis of matched pairs.
8. Matched Pair Testing

Figure 8.5: Three matched pairs have been added to the configuration in Figure 8.4.

8.6 Numerical Experiments

We performed four numerical experiments designed to explore the robustness of 3WNM to changes in $\delta_1$ and $\delta_2$. Sections 8.6.1 and 8.6.2 describe simulation experiments in which $\Xi_1 = \Xi_2$. Section 8.6.3 describes an example of disparate feature representations. Section 8.6.4 describes an experiment matching faces in images observed under two conditions.

8.6.1 Gaussian Simulation

Here we extend the illustrative example. We chose $\delta_1$ to measure Euclidean distance and considered three choices of $\delta_2$: Euclidean distance, squared Euclidean distance, and “city block” ($L_1$) distance. For each choice of $\delta_2$ we performed 100 replications of the following experiment.

First, using the probability model described in Section 8.5, we generated $N = 20$ training pairs,
8. Matched Pair Testing

Figure 8.6: Three mismatched pairs have been added to the configuration in Figure 8.4.

(x_{i1}, x_{i2}), and 200 test pairs, (y_{i1}, y_{i2}). Of the 200 test pairs, 100 were matched and 100 were mismatched. Second, we constructed representations of the data in \( \mathbb{R}^2 \) using the 3WNM and JOFC procedures. Third, for each representation we estimated the tradeoff between significance level (\( \alpha \)) and power (\( \beta \)) by (a) using the empirical distribution of \( d(\tilde{y}_1, \tilde{y}_2) \) for the matched pairs to determine a critical value \( c_\alpha \) for each achievable \( \alpha \), then (b) estimating the power of the level-\( \alpha \) test by the proportion of mismatched pairs for which \( d(\tilde{y}_1, \tilde{y}_2) > c_\alpha \).

Figures 8.9, 8.10, and 8.11 plot \((\alpha, \beta)\), averaged over 100 replications, for the three choices of \( \delta_2 \). Because 3WNM is nonmetric and squaring is a monotonic transformation, any differences between 3WNM’s \((\alpha, \beta)\) curves in Figures 8.9 and 8.10 is due to chance variation. In contrast, Euclidean and city block distances are not monotonically related, resulting in an intrinsically different \((\alpha, \beta)\) curve. The 3WNM curves are slightly above (more powerful) the JOFC curves, and slightly more so for \( \delta_2 \neq \delta_1 \).
8. Matched Pair Testing

8.6.2 Dirichlet Simulation

[67] used Dirichlet distributions to model the pairing of documents written in different languages. Accordingly, we repeated the experiments described in Section 8.6.1 using Dirichlet distributions instead of normal distributions and drawing $\omega_i \sim \text{Dirichlet}(1, 1, 1)$, $s_{ik} \sim \text{Dirichlet}(30\omega_i + 1)$, and $\epsilon_{ik} \sim \text{Dirichlet}(1, 1, 1)$. Figures 8.12, 8.13, and 8.14 display $(\alpha, \beta)$ curves for 3WNM and JOFC for the same three choices of $\delta_2$, with patterns comparable to Figures 8.9, 8.10, and 8.11.

8.6.3 Wikipedia

[67] compared English and French Wikipedia articles related to algebraic topology. Here we compare two disparate representations of the English articles. The objects are the $N = 1382$ articles in the directed two neighborhood of the “Algebraic Topology” node of the English Wikipedia link graph. We represented each article as (1) a node in the link graph of the extracted articles, and (2)
8. Matched Pair Testing

Figure 8.8: Kernel density estimates of the test statistic under the matched and mismatched conditions.

a point in a vector space model of article content. The corresponding dissimilarity measures were
(1) the shortest path distance in the link graph, and (2) the cosine dissimilarity of the discounted
mutual information [67, 52, 62]. Here $\delta_1$ and $\delta_2$ are completely disparate measures of dissimilarity,
a circumstance for which 3WNM was designed but JOFC was not. Note that the problem of
determining whether or not an article’s content matches its location in the link graph mimics the
problem of identifying spam websites on the world wide web.

Because we lacked a probability model from which to generate additional matched and un-
matched pairs, we used five-fold cross-validation to estimate the relationship between $\alpha$ and $\beta$. We
partitioned the $N = 1382$ articles into $F_0, \ldots, F_4$, then repeated the following for $i = 0$ to 4:

1. Let $F_i$ denote the training set.

2. Let $F_{(i+1) \text{ mod } 5}$ denote a set of matched test pairs.
8. Matched Pair Testing

Figure 8.9: (Gaussian Setting) δ₁ and δ₂ are both Euclidean distance.

3. Let respective pairs in $F_{(i+2) \mod 5}$ and $F_{(i+3) \mod 5}$ be used as mismatched test pairs.

Both 3WNM and JOFC embedded in $\mathbb{R}^{10}$. The $(\alpha, \beta)$ curves were estimated by averaging $\beta$ over the five testing scenarios.

Figure 8.15 shows the estimated $(\alpha, \beta)$ curves for 3WNM and JOFC. The JOFC approach was not designed for disparate measures of dissimilarity and displays almost no power. The 3WNM approach, which ignores metric information and only considers rank information, displays considerably greater power.

8.6.4 Face Matching

We demonstrate our matched pair testing approach using a dataset consisting of eight photos of 25 people taken under each of two conditions (“controlled” and “uncontrolled” environments).
In the context of this experiment, a matched pair consists of a “controlled” photo and an “uncontrolled” photo of the same person’s face. We split the data into sets of 15 people for training and 10 people for testing.

Each image is represented by 100 eigenface features. We ranked the eigenface features in both the “controlled” and “uncontrolled” training set photos according to separation of the matched and mismatched pairwise distances using the two-sample t-statistic. Next, we choose the number of features to use to compute distances. Using a hold out set from the training set we determined that two features maximizes the area under the ROC curve.

We used 3WNM to embed the training set into a reference configuration in \( \mathbb{R}^4 \). We out-of-sample embed the test pairs and compute the distances for matched and mismatched pairs. Figure 8.16 shows the empirical CDFs of the distances. Figure 8.17 shows an estimated \((\alpha, \beta)\) curve.

Preliminary experimentation highlighted the benefit of feature selection in the face matching
8. Matched Pair Testing

Figure 8.11: (Gaussian Setting) $\delta_1$ is Euclidean distance and $\delta_2$ is city block distance.

experiment. It was apparent that feature selection significantly increases statistical power. In the context of the 3WNM approach, this experiment reinforced the importance of using high quality measures of dissimilarity.

8.7 Discussion

[67] proposed a novel approach to the problem of matched pair hypothesis testing. We have modified their ideas for use with disparate dissimilarity measures. Nonmetric MDS allows monotonic transformations of the input dissimilarities, thereby using the ranks of the dissimilarities but not their numeric values.

Although we might have used any of several nonmetric embedding methodologies, we adopted a three-way approach because of the relation between the so-called identity model for three-way
MDS and the JOFC formulation proposed in [67]. Other formulations may also have merit. Our numerical experiments suggest that using three-way nonmetric MDS approach provides good results both when the dissimilarity measures are similar and when they are disparate.
8. Matched Pair Testing

Figure 8.13: (Dirichlet Setting) $\delta_1$ is Euclidean distance and $\delta_2$ is squared Euclidean distance.

Figure 8.14: (Dirichlet Setting) $\delta_1$ is Euclidean distance and $\delta_2$ is city block distance.
8. Matched Pair Testing

Figure 8.15: Wikipedia Experiment

Figure 8.16: Empirical CDF of the test statistic under the matched and mismatched conditions.
Figure 8.17: Face Experiment (AUC = 0.8209)
9

Embedding Sparse Ordinal Graphs

9.1 Introduction

A number of interesting data sets provide information about the nearest neighbors of a set of objects. For example, each book sold by Amazon.com is accompanied by a list of its most similar neighbors, measured by the proportion of buyers who ultimately also buy the other book. Likewise, entities such as Netflix and internet radio site Last.fm make recommendations based on nearest neighbors, determined in various ways. In many such applications, information is available (or reliable) for only a small set of nearest neighbors; furthermore, information about proximity may be ordinal, or measured on a scale that necessitates an unspecified monotonic transformation.

The data sets of interest can be conceived as directed graphs in which vertices represent objects and edges identify nearest neighbor structure. In order to visualize such data, we consider the problem of how to represent a sparse directed graph with ordinal edge weights in a low-dimensional Euclidean space. Such problems are the concern of graph layout and/or graph embedding, distinguished by the latter’s greater concern that Euclidean distances between points in the representation should approximate pairwise dissimilarities of objects.
The problem of embedding graphs that describe local structure arises elsewhere. In computational chemistry, one may attempt to determine a molecule’s 3-dimensional structure from information about its atomic bond lengths and angles. In manifold learning, techniques such as ISOMAP [81] and Laplacian eigenmaps [3] construct and embed graphs that describe the local Euclidean structure of curved surfaces. The edges of these (undirected) graphs may or may not be weighted, but the problem of embedding graphs with ordinal edge weights has not been considered.

In statistics and psychometrics, techniques for embedding pairwise proximity data are called multidimensional scaling (MDS) and techniques that use only ordinal information about the proximities are called nonmetric MDS. There is a vast literature on nonmetric MDS, but nonmetric MDS usually embeds a complete set of pairwise proximity ranks. In contrast, scant attention has been paid to problems in which only local proximity information is available. A recent exception is the Local MDS method of Chen and Buja [12], inspired in part by algorithms for graph layout. Although Local MDS assumes the existence of metric proximities, it most nearly embodies the ideas described herein.

9.2 Embedding

Consider a directed graph $G = (V, E)$ with vertices $V = \{1, \ldots, N\}$. For each vertex $i$, assume that the edges $(i, j) \in E$ can be ranked from smallest to largest and let $r_{ij}$ denote the rank of edge $(i, j)$. If $r_{ij_1} = 1$, then we say that vertex $j_1$ is the nearest neighbor of vertex $i$; if $r_{ij_2} = 2$, then we say that vertex $j_2$ is the second nearest neighbor of vertex $i$; etc. For simplicity, we assume that all ties have been broken. For convenience, we then set $r_{ij} = \infty$ if $(i, j) \notin E$.

Let $d$ denote the dimension of the Euclidean space in which we seek to embed $G$. Ideally, we would like to find a set of points $x_1, \ldots, x_N \in \mathbb{R}^d$ with the following property:
(I) If $r_{ij} < r_{ik}$, then $d_{ij} = \|x_i - x_j\| \leq \|x_i - x_k\| = d_{ik}$.

Henceforth, we denote a possible configuration of $N$ points in $\mathbb{R}^d$ by $X$ and the interpoint distances for $X$ by $d_{ij}(X)$. We would like to find an $X$ whose interpoint distances satisfy a prescribed partial ordering.

Several considerations affect our formulation of the above embedding problem.

1. **Existence**  In practice, it is often the case that no $X$ has interpoint distances for which the desired partial ordering is fully satisfied. This consideration obliges us to formulate an error criterion that quantifies the suitability of each $X$, then search for $X$ that minimize the error criterion.

2. **Asymmetric Information**  Euclidean distances are necessarily symmetric: $d_{ij}(X) = d_{ji}(X)$. In contrast, nearest neighbor graphs are typically asymmetric, i.e., one usually finds that $r_{ij} \neq r_{ji}$. A reasonable error criterion should accommodate this asymmetry.

3. **Local Structure**  The graphs that we are attempting to embed are sparsely connected. In particular, we only have information about a small number of nearest neighbors for each object. Graef and Spence [35] demonstrated that the pattern of sparsity plays a crucial role in determining the quality of an embedding. In fact, their simulations indicate that nearest neighbor proximities are less important than non-neighbor proximities for determining global structure. (This revelation is hardly surprising: consider, for example, that the chemical structure of a molecule may be consistent with a number of different 3-dimensional structures.) In consequence, it is important to use what little global information can be inferred, viz., that non-neighbors of vertex $i$ are less like $i$ than are neighbors of $i$. Nevertheless, we should not expect too much from any embedding method. As with graph layout, a variety of configurations may be deemed of comparable quality.
9. *Embedding Sparse Ordinal Graphs*

4. **Ordinal Constraints**  Traditional formulations of nonmetric MDS, including Kruskal’s [45] seminal contribution, incorporate a complete ordering of \(N(N - 1)/2\) pairwise proximities into a scale-invariant error criterion. The cost of isotonic regression is \(O(N^2)\). Scale invariance precludes degenerate solutions. In contrast, Trosset [84] demonstrated that the objectives of nonmetric MDS can be achieved by enforcing explicit order and nondegeneracy constraints. For general partial orderings, the cost of Burdakov’s [9] algorithm for isotonic regression is \(O(N^4)\).

5. **Scalable Algorithms**  As with Amazon.com, nearest neighbor graphs are especially popular when \(N\) is quite large. Nonmetric MDS was developed for situations in which \(N\) is fairly small, no more than several hundred. To embed much larger graphs with ordinal weights, we combine several ideas. We construct inexpensive initial configurations. We adopt an error criterion that can be decreased by an inexpensive fixed point method, thereby avoiding the usual costs of numerical optimization. We use specific partial orderings that allow us to use fast algorithms for isotonic regression. We stop after a fixed number of iterations, without monitoring convergence to a minimizer of our error criterion.

9.3 **Optimization**

We seek a configuration that, in some sense, captures the local structure of a nearest neighbor graph. Often, the only configurations that satisfy property (I) are degenerate in the sense that they place each point at the same location, so that each \(d_{ij}(X) = 0\). We might search for a nondegenerate \(X\) that violates property (I) as rarely as possible, but the resulting optimization problem seems intractable. Instead of counting how often the prescribed partial ordering is violated, we measure the magnitude of the violations.
Given $x_1, \ldots, x_N \in \mathbb{R}^d$ and $j \neq i$, let $r_{ij}(X)$ denote the rank of $d_{ij}(X)$ in $\{d_{ik}(X) : k \neq i\}$. For example, if $x_j$ is $x_i$’s second nearest neighbor, then $r_{ij}(X) = 2$. It is natural to seek small values of the following error criterion:

$$M_1(X) = \frac{1}{|E|} \sum_{(i,j) \in E} |r_{ij}(X) - r_{ij}|.$$  

Alternatively, Chen and Buja [12] counted how many nearest neighbor pairs in $G$ are still nearest neighbor pairs in $X$. Let $E_k(X)$ denote the edges of the nearest neighbor graph constructed from $X$. Then, after scaling by the number of edges in $G$, the Chen-Buja criterion is

$$M_2(X) = \frac{|E_k(X) \cap E|}{|E|}.$$  

This criterion seems most appropriate when $G$ is unweighted.

Both $M_1$ and $M_2$ are intuitively appealing error criteria, but neither appears to be amenable to direct optimization. Although we will monitor these quantities, our searches will be guided by solving a more tractable optimization problem that was inspired by certain formulations of nonmetric MDS.

Recall that a matrix, $\Delta = [\delta_{ij}]$, is nonnegative iff each entry $\delta_{ij} \geq 0$ and hollow iff each diagonal entry $\delta_{ii} = 0$. Let $\mathcal{M}(G)$ denote the polyhedral cone of hollow nonnegative matrices that satisfy the partial ordering induced by $G$, i.e., if $r_{ij} < r_{ik}$, then $\delta_{ij} \leq \delta_{ik}$.

Given $x_1, \ldots, x_N \in \mathbb{R}^d$, let $D(X) = [d_{ij}(X)]$. We seek an $X$ for which $D(X)$ is near $\mathcal{M}(G)$; however, because $0 \in \mathcal{M}(G)$, we must be careful to avoid degenerate solutions. One possible
9. Embedding Sparse Ordinal Graphs

antidote is as follows. For some \( c > 0 \), let

\[
N = \left\{ \Delta : \sum_i \sum_j \delta_{ij}^2 \geq c \right\}
\]

and seek an \( X \) for which \( D(X) \) is near \( M(G) \cap N \).

We measure the discrepancy between \( D(X) \) and \( \Delta \in M(G) \cap N \) by an asymmetrically weighted version of Kruskal’s [45] raw stress criterion:

\[
\sigma_a(\Delta, X) = \frac{1}{2} \sum_i \sum_j w_{ij} \left[ d_{ij}(X) - \delta_{ij} \right]^2.
\]

We emphasize that we require neither \( \delta_{ij} = \delta_{ji} \) nor \( w_{ij} = w_{ji} \).

The choice of weights is crucial. We restrict attention to \( w_{ij} \in \{0, 1\} \) and consider three possible weighting schemes.

1. \( w_{ij} = 1 \) iff \( (i, j) \in E \)

This choice corresponds to using exactly the information contained in the nearest neighbor graph. Although it is conceptually attractive, the information is so limited that one rarely obtains a satisfying configuration. In numerical experiments with metric MDS, Graef and Spence [35] found that such schemes often lead to undesirable configurations. In our experience, the problem is just as acute with ordinal information. The difficulty is that local information is rarely sufficient to infer a plausible global structure.

2. \( w_{ij} = 1 \) for all \( i, j \)

This choice is the usual raw stress criterion. In the present context, it corresponds to using all of the information contained in the nearest neighbor graph plus all of the additional information that can be inferred from it, i.e.,
if \((i, j) \notin E\) and \((i, k) \in E\), then \(\delta_{ij} \geq \delta_{ik}\).

The problem with this choice is that of too much of a good thing. Including so many non-edge pairs causes \(\sigma_a\) to undervalue the edge; furthermore, for large \(N\), the expense of embedding \(N(N-1)/2\) pairs may be prohibitive.

3. \(w_{ij} = 1\) for all \((i, j) \in E\) and some additional \((i, j) \notin E\)

This choice is a compromise between the first two. We randomly select a small set of non-edge pairs and include them in \(\sigma_a\). The additional information tends to produce more plausible configurations than are typically obtained from just the edge pairs. We denote the augmented set of pairs by \(E^+\).

Combining the above, we obtain the following optimization problem in the decision variables \((\Delta, X)\):

\[
\begin{align*}
\text{minimize} & \quad \sigma_a(\Delta, X) \\
\text{subject to} & \quad \Delta \in \mathcal{M}(G) \cap \mathcal{N}
\end{align*}
\]

Although (9.1) is a tractable optimization problem and algorithms for solving it tend to produce plausible embeddings, we emphasize that configurations with vastly different local structure may have comparable values of \(\sigma_a\). Consider Figure 9.1, which caricatures the problem of inferring molecular conformation from chemical structure. Within the subsets defined by shape/color, the two configurations have identical local structure. The global structure is evidently quite different, as is the local structure between the subsets. If \(w_{ij} = 1\) for pairs within the same subset and \(w_{ij} = 0\) for pairs between subsets, then \(\sigma_a\) cannot distinguish these configurations. Notice, however, that these configurations do have different values of \(M_1\) and \(M_2\). We will further explore
9. Embedding Sparse Ordinal Graphs

the relationship between these error criteria in Section 9.5.2.

![Figure 9.1: Identical \( \sigma_a \), different \( M_1 \) and \( M_2 \).](image)

9.4 The NNScal Algorithm

Given \((\Delta_c, X_c)\), we can find a better \((\Delta_+, X_+)\) by first fixing \(\Delta_c\) and finding a better \(X_+\), then fixing \(X_+\) and finding a better \(\Delta_+\). Doing so is an example of variable alternation. To adopt this strategy, it is necessary to construct an initial \((\Delta, X)\). We construct an initial \(\Delta = [\delta_{ij}]\) by fixing \(\hat{r} > \max(\{r_{ij} : r_{ij} < \infty\})\), then setting \(\delta_{ij} = \min(r_{ij}, \hat{r})\). To construct an initial \(X\), we first symmetrize \(\Delta\), setting \(\delta_{ij} = \delta_{ji} = \min(\delta_{ij}, \delta_{ji})\), then embed \(\Delta_{sym}\) by metric MDS to obtain an initial \(X\). Here is the algorithm:
9. Embedding Sparse Ordinal Graphs

Algorithm 9.1. NNScal (Nearest Neighbor SCALing):

1. Specify \( \hat{r} \), e.g., \( \hat{r} = 2 \max(\{r_{ij} : r_{ij} < \infty\}) \).

2. Set \( \Delta = \min(r_{ij}, \hat{r}) \) and \( \Delta_{\text{sym}} = \min(r_{ij}, r_{ji}, \hat{r}) \).

3. Construct an initial configuration \( X \) by embedding \( \Delta_{\text{sym}} \).

4. Repeat until convergence:

   (a) Fix \( \Delta \) and minimize \( \sigma_a \) with respect to \( X \) by asymmetric stress Majorization.

   (b) Fix \( X \) and minimize \( \sigma_a \) with respect to \( \Delta \in \mathcal{M}(G) \) by isotonic regression.

   (c) Scale \( \Delta \) to lie in \( \mathcal{M}(G) \cap N \).

As stated, the NNScal algorithm converges to stationary points of (9.1). In practice, we only perform several iterations of asymmetric stress majorization, obtaining an improved but suboptimal \( X \). Furthermore, we do not repeat the variable alternation cycle until convergence. Several iterations usually suffice to obtain a reasonable embedding, and that may be all that we can afford if \( G \) is large.

The remainder of this section explicates various details of NNScal.

9.4.1 Initial Configuration

The objective function \( \sigma_a \) is not convex with respect to \( (\Delta, X) \). It may have many nonglobal minimizers; hence, the quality of the embedding returned by NNScal is sensitive to the choice of initial \((\Delta, X) \). Given \( \Delta_{\text{sym}} \), one might construct \( X \) by classical MDS. This is a common way to construct an initial configuration from which to begin minimizing the raw stress criterion, but it requires computing the spectral decomposition of an \( N \times N \) matrix. This computation is prohibitively
9. Embedding Sparse Ordinal Graphs

expensive when $N$ is large. One can dramatically decrease the computational expense of the initial embedding through techniques such as Landmark MDS [23], FastMap [30], and Metric Map [91], each of which can be viewed as a crude Nyström approximation of the eigenvalues and eigenvectors used in classical MDS [65]. However, these techniques may produce configurations of low quality and they do not exploit the special structure of nearest neighbor graphs. We leave the problem of how to do so for future research. In Section 9.5, we start NNScal with initial configurations obtained by classical MDS with imputed $\hat{r}$ (IC1) and generated at random (IC2).

9.4.2 Asymmetric Stress Majorization

Once a configuration has been obtained, Step 4a improves upon it by fixing $\Delta$ and minimizing $\sigma_a$ with respect to $X$.

For $i < j$, define square matrices $A_{ij}$ by setting the $ii$ and $jj$ entries equal to 1, the $ij$ and $ji$ entries equal to $-1$, and all other entries equal to 0. Kruskal’s raw stress criterion,

\[
\sigma(\Delta, X) = \sum_{i < j} w_{ij}(\delta_{ij} - d_{ij}(X))^2
= \eta_3^2 + \text{tr} (X^T V X) - 2\text{tr} (X^T B(X) X),
\]

where $V = \sum_{i<j} w_{ij}A_{ij}$ and $B(X) = \sum_{i<j} \frac{w_{ij}\delta_{ij}}{d_{ij}} A_{ij}$, was intended for use with symetric $\Delta$. We seek to minimize the asymmetric raw stress criterion:

\[
\sigma_a(\Delta, X) = \frac{1}{2} \sum_i \sum_j w_{ij}(\delta_{ij} - d_{ij}(X))^2
= \frac{1}{2} \left[ \eta_3^2 + \text{tr} (X^T V_a X) - 2\text{tr} (X^T B_a(X) X) \right],
\]
9. Embedding Sparse Ordinal Graphs

where \( V_a = \sum_{i<j} (w_{ij} + w_{ji}) A_{ij} \) and \( B_a(X) = \sum_{i<j} \frac{w_{ij} \delta_{ij} + w_{ji} \delta_{ji}}{d_{ij}} A_{ij} \).

For fixed \( \Delta \), both criteria can be majorized by functions of the same form. For \( \sigma_a \), the Diagonal Majorization Algorithm (DMA) of Trosset and Groenen [88] updates the configuration as follows:

\[
X \leftarrow X + \frac{1}{2} \text{diag}(V_a)^{-1} [B_a(X) - V_a] X.
\]

For nearest neighbor graphs, the update can also be computed as

\[
x_i \leftarrow x_i + \frac{1}{2\gamma(i)} \left[ \sum_{(i,j) \in E^+} \left( \left( \frac{\delta_{ij}}{d_{ij}} - 1 \right) (x_i - x_j) \right) + \sum_{(j,i) \in E^+} \left( \left( \frac{\delta_{ij}}{d_{ij}} - 1 \right) (x_j - x_i) \right) \right],
\]

where \( \gamma(i) \) is the in-degree plus the out-degree of vertex \( v_i \). The second update formula is evaluated by looping through the edges in \( E^+ \) and updating \( x_i \) for each \( (i, j) \) and \( (j, i) \) pair. The first update formula requires \( O(N^2d) \) operations, whereas the second update formula requires only \( O(Nkd) \) operations.

For fixed \( \Delta \), repeated iterations of DMA will converge to a stationary configuration of \( \sigma_a \). In practice, we do not monitor a convergence criterion in Step 4a, but simply perform a fixed number of iterations (typically 4 or 5) before proceeding to Step 4b.

9.4.3 Isotonic Regression

Step 4b of NNScal fixes \( X \) and updates \( \Delta \) subject to \( \Delta \in \mathcal{M}(G) \). The unique minimizer of this constrained least squares problem can be computed efficiently. The partial ordering induced by the constraints is a set of \( N \) distinct, complete orderings. Each of the \( N \) subproblems has a predefined rank ordering for the nearest neighbors and no predefined ranking for the additional pairs in \( E^+ - E \). Without loss of generality, consider the case where \( i = N \) and \( v_i \)'s \( k \) nearest
neighbors, in order, are \( v_1, v_2, \ldots, v_k \). There is also an additional set of \( M \) edges in \( E^+ \) that are directed edges from \( v_i \) to \( v_{k+1}, v_{k+2}, \ldots, v_{k+M} \). Then, for a fixed \( X \) where \( D(X) = [d_{ij}] \), we seek solutions of:

\[
\begin{align*}
\min_{\delta_{i1}, \ldots, \delta_{ik+M}} & \quad \frac{1}{2} \sum_{j=1}^{k+M} w_{ij} (\delta_{ij} - d_{ij})^2 \\
\text{s.t.} & \quad \delta_{i1} \leq \delta_{i2} \leq \ldots \leq \delta_{ik} \preceq \{\delta_{n,k+1}, \delta_{n,k+2}, \ldots, \delta_{n,k+M}\}.
\end{align*}
\]

Here, the \( \preceq \) means that \( \delta_{ik} \) is entry-wise less than each element of the succeeding set.

To perform this optimization problem we first sort the non-neighbor distances from the configuration \( X \) with non-zeros weights. Again, without loss of generality let us assume that the distances are \( d_{i,k+1}, \ldots, d_{i,k+M} \) and satisfy \( d_{i,k+1} \leq d_{i,k+2} \leq \ldots \leq d_{i,k+M} \). Then we can solve the completely ordered isotonic regression problem:

\[
\begin{align*}
\min_{\delta_{i1}, \ldots, \delta_{i,k+M}} & \quad \frac{1}{2} \sum_{j=1}^{k+M} w_{ij} (\delta_{ij} - d_{ij})^2 \\
\text{s.t.} & \quad \delta_{i1} \leq \delta_{i2} \leq \ldots \leq \delta_{i,k+M}.
\end{align*}
\]

We perform \( N \) completely ordered isotonic regressions, one for each \( i \), using the algorithm in [36]. Since each isotonic regression requires a sort to determine the ordinal constraints with an average case computation time \( O(M \log M) \), an isotonic regression that requires \( O(k + M) \) time, and it is assumed that \( M \in \Omega(k) \) then the resulting average case computation time is \( O(NM \log M) \).

### 9.4.4 Precluding Degenerate Solutions

Without the constraint that \( \Delta \in \mathcal{N} \) in (9.1) an optimal solution would be \( \Delta = 0 \) and \( X = 0 \). Clearly, this solution is undesirable. In [84], it was proposed that one might preclude such
9. Embedding Sparse Ordinal Graphs

degenerate solutions by bounding the sum of the squared dissimilarities from below. We denote
the updated dissimilarities from the isotonic regression step (NNScal 4b) by $\Delta = [\delta_{ij}]$. Next, we
compute $\tau = \sum_i \sum_j w_{ij} \delta_{ij}$. By lemma 2 in [84], we satisfy the nondegeneracy constraint if we
update the dissimilarities as follows: iff $\tau < c$, we rescale the dissimilarities $\delta_{ij} \leftarrow c \delta_{ij}/\tau$.

9.5 Numerical Experiments

In this section, we perform numerical experiments to demonstrate NNScal. In 9.5.1, we exhibit
the effect of the sparsity of the nearest neighbor graph. In 9.5.2, we demonstrate the relationship
between $\sigma_a$, $M_1$, and $M_2$. In 9.5.3, we demonstrate its performance on a large, sparse nearest
neighbor graph ($N = 50,000$).

9.5.1 The effect of sparsity

Inspired by the experimentation in [35], we constructed 10 unique configurations of 200 points
uniformly drawn from the unit circle in $\mathbb{R}^2$. We use the $k$-nearest neighbor graphs according to
Euclidean distance in $\mathbb{R}^2$. Configurations were used only if their 4-nearest neighbor graph was
connected. Graef and Spence [35] were interested in studying the effect of missing proximities on
metric MDS given perturbed interpoint distances. In our formulation, no metric information is
known, however, we know the ordering of the neighbor proximities.

Two initial configurations were generated, one by Classical MDS with $\hat{r} = 2 \max\{r_{ij} : r_{ij} < \infty\}$ (IC1) and one at where $x_1, \ldots, x_N \sim \text{Normal}(0, I_d)$ (IC2). In practice, generating an initial con-
figuration with Classical MDS is not always feasible due to computational expense. This example
demostrates the importance of the initial configuration.
We performed two separate experiments. First, for each initial configuration and dataset pair we construct a $k$-nearest neighbor graph and add $m = 1$ non-neighbor edge. Second, we construct a $k$ nearest neighbor graph and add $m = 5k$ non-neighbor edges. In each of these trials $k \in \{10, 13, 16, 19, 22, 25, 28\}$. We plot the mean $M_1$ and $M_2$ of the ten datasets for each choice of $k$ in Figure 9.2.

The importance of a quality initial configuration is evident in all four plots. Also, the inclusion of additional proximities had a negative effect for small $k$ and negligible effect for large $k$.

9.5.2 Stress vs. $M_1$ and $M_2$

In this section, we study the effect of minimizing stress as a heuristic for minimizing $M_1$ and $M_2$. We use one dataset from above and construct its $k = 20$-nearest neighbor graph with $5k$ non-neighbor edges included. For this example, we start with a random initial configuration, repeat step 4 of NNScal 50 times, and iterate DMA 4 times for each step 4a. We plot $\log(\sigma_a)$, $M_1$, and $M_2$ (Figure 9.3) for each iteration. Each criterion was computed after each update of either $X$ or $\Delta$.

Evidently, the relationship between asymmetric stress, $M_1$, and $M_2$ tends to be monotonic. The plots indicate that $M_1$ and $M_2$ do not appreciably improve until stress has been significantly decreased. We then repeated the above experiment, but instead of using a randomly generated configuration (IC2) we start with a more principled configuration (IC1) (see Figure 9.4). This experiment demonstrates the importance of the initial configuration on the ability to improve $M_1$ and $M_2$.

We plot the embedding obtained starting from an initial random configuration (IC2) in the top left plot of Figure 9.5. We added line segments representing the original edges in the nearest neighbor graph (top right plot in Figure 9.5). To remove visual clutter we picked two sets of 5 points at random and added the edges to their respective nearest neighbors (bottom left and bottom right
plots in Figure 9.5). It is evident from these plots that neighbor structure was reasonably well maintained in the embedding.

### 9.5.3 “Large Graph”

We generated a random, connected, 9-nearest neighbor graph containing $|V| = 50,000$ nodes. For each node, $M = 5k = 45$ non-neighbors were selected at random to augment the edge set. We repeated Step 4 of NNScal four times and step 4a was repeated three times per iteration. The purpose of this experiment is to demonstrate that we can minimize asymmetric stress on large proximity graphs. We did the computation on a single core of a 2.4 GHz Intel Core 2 Duo. In Figure 9.6, we plot the log of the asymmetric stress with respect to wall clock time.
Figure 9.2: Solid line = IC1, Dashed line = IC2 (with ±1 standard deviation error bars)
Figure 9.3: Asymmetric stress (top left), $M_1$ (top right), $M_2$ (bottom) – Randomly generated configuration (IC2)
9. Embedding Sparse Ordinal Graphs

Figure 9.4: Asymmetric stress (top left), $M_1$ (top right), $M_2$ (bottom) – Classical MDS initial configuration (IC1)
Figure 9.5: Embedding (top left) with edges drawn to connect nearest neighbors (other three plots).
Figure 9.6: $\log(\sigma_a)$ on a large graph (n=50,000)
Bibliography


