2013

Information and distances

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

INFORMATION AND DISTANCES

by

SAMUEL EPSTEIN

B.S., Georgia Institute of Technology, 2003
M.S., Boston University, 2013

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2013

Approved by

First Reader

Leonid A. Levin, PhD
Professor of Computer Science

Second Reader

Margrit Betke, PhD
Professor of Computer Science

Third Reader

Benjamin Hescott, PhD
Assistant Professor of Computer Science
Tufts University

Acknowledgments

I would like to thank Michelle Rheaume for her continued support.

INFORMATION AND DISTANCES

(Order No.)

SAMUEL EPSTEIN

Boston University, Graduate School of Arts and Sciences, 2013

Major Professor: Leonid A. Levin, PhD,
Computer Science

ABSTRACT

We prove all randomized sampling methods produce outliers. Given a computable measure P over natural numbers or infinite binary sequences, there is no method that can produce an arbitrarily large sample such that all its members are typical of P. The second part of this dissertation describes a computationally inexpensive method to approximate Hilbertian distances. This method combines the semi-least squares inverse techinque with the canonical modern machine learning technique known as the kernel trick. In the task of distance approximation, our method was shown to be comparable in performance to a solution employing the Nyström method. Using the kernel semi-least squares method, we developed and incorporated the Kernel-Subset-Tracker into the Camera Mouse, a video-based mouse replacement software for people with movement disabilities. The Kernel-Subset-Tracker is an exemplar-based method that uses a training set of representative images to produce online templates for positional tracking. Our experiments with test subjects show that augmenting the Camera Mouse with the Kernel-Subset-Tracker improves communication bandwidth statistically significantly.

Contents

List of Tables

List of Figures

List of Abbreviations

AIT	.....	Algorithmic Information Theory
HCI	.....	Human Computer Interaction
NRI	.....	Normalized Radial Intensity
PIE	.....	CMU Pose, Illumination, and Expression Database

Chapter 1

Introduction

This thesis comprises theoretical and applied studies in inference, i.e. the extrapolation of the hidden part of the environment from a collection of observations. The theoretical part of the thesis focuses on algorithmic information theory (AIT). One central construct of AIT is Kolmogorov complexity, $\mathbf{K}(b)$, the size of the shortest binary program that outputs a natural number b on a universal self-delimiting Turing machine. This work was initiated with the study of two-player games in the context of AIT [Epstein and Betke (2011a)].

We use the concept of the mutual information of a number b with the halting sequence $\mathcal{H}$, the characteristic sequence of the domain of a universal Turing machine. We call ***exotic*** numbers (and any object they encode) that have high mutual information with $\mathcal{H}$. Assuming the independence postulate [Levin (2002)], there is no method to generate *exotic* numbers. The present thesis is concerned with properties restricted to *non-exotic* numbers.

The thesis also contains applied results in inference. Using generalized matrix inversion techniques, we introduce the ***Kernel Semi-least Squares*** method. This method can be used to approximate a computationally expensive metric using a set of offline calculations. This method was incorporated into the Camera Mouse, a video-based mouse replacement system for people with disabilities.

1.1 Games

Chapter 3 details the work in [Epstein and Betke (2011a)], where algorithmic information theory was used to provide insights into modeling information between agents of two-player games. Players were formally represented using finite sets of strings A and B. Each string in A encodes an action of player 2 and player 1's best response (and similarly for set B). Each string $x \in A \cap B$ represents an equilibrium between the players A and B. The work in [Epstein and Betke (2011a)] introduced an application of a lemma in [Vereshchagin and Vitányi (2004a),Vereshchagin and Vitányi (2010)].

1.2 Complexity of Classification

Chapter 4 contains upper bounds on the minimal encoding length of a predicate (over the set $\mathbb{N}$ of natural numbers) consistent with a predicate γ over a finite domain $D \subset \mathbb{N}$. For *non-exotic* predicates γ, the least encoding length of a complete predicate consistent with γ is close to the size of D. This is in contrast with the existence of predicates with 2-point domains and arbitrary high minimum complexity of complete predicates consistent with them. Thus the minimal complexity of a *non-exotic* complete extension of γ is small whereas the complexity of encoding all the pairs $\langle b, \gamma(b) \rangle$ of γ could be much larger due to unlimited bit length of b. Thus, there could be no methods to produce such samples.

1.3 Non-Exotic Samples

In many areas, one needs to obtain numbers $b \in \mathbb{N}$ that are "typical" with respect to a computable measure P over $\mathbb{N}$. Being typical with respect to P means having a small ***deficiency of randomness***, $\mathbf{d}(b|P) \stackrel{\text{def}}{=} \|P(b)\| - \mathbf{K}(b)$. Atypical elements b with large $\mathbf{d}(b|P) \gg 0$ have extra regularities that allow them to be compressed to

length $\mathbf{K}(b)$, which is much less than $\|P(b)\|$. With a little luck, it is not difficult to produce elements b with $\mathbf{d}(b|P) \approx 0$, or small sets of such elements. However, we show in chapter 4 that only exotic samples $D \subset \mathbb{N}$ of large size $\|D\| = 2^n$ have all elements $b \in D$ typical of P, with $\mathbf{d}(b|P) = o(n)$. There could be no methods to produce such samples.

One of the issues in the application of Algorithmic Information Theory is the uncomputability of Kolmogorov complexity. There does not even exist an algorithm that computes any non-constant lower bound of $\mathbf{K}$. In chapter 4, we show that every encoding of 2^n unique pairs $\langle b, \mathbf{K}(b) \rangle$ has more than $\sim n$ bits of mutual information with the halting sequence $\mathcal{H}$. Thus, all such large sets are *exotic*.

1.4 Kernel Semi-least Squares

Chapter 5 details a distance approximation method that uses generalized matrix inverse techniques developed in the early 1970s, as seen in [Epstein and Betke (2011b)]. The work in [Radhakrishna and Mitra (1971)] introduced the notion of a semi-least squares inverse to the semi-least squares problem. This semi-least squares inverse methodology can be combined with the canonical modern machine learning technique known as the kernel trick [Schölkopf and Smola (2001)]. This technique can be leveraged to create an efficient method for approximating a computationally expensive distance of an object to a large number of other elements. This ***Kernel Semi-least Squares*** method is useful for the task of distance approximation and was tested against competitor methods using two independent databases of images [Graham and Allinson (1998); Sim et al. (2002)].

1.5 Tracking with the Camera Mouse

Chapter 6 details the incorporation of the ***Kernel Semi-least Squares*** method into the Camera Mouse, a video-based mouse replacement system for people with disabilities, as seen in [Epstein et al. (2012)]. Some people cannot use their hands to control a computer mouse due to conditions such as cerebral palsy or multiple sclerosis. For these individuals, there are various mouse-replacement solutions. One approach is to enable them to control the mouse pointer using head motions captured with a web camera. One such system, the Camera Mouse, uses an optical flow approach to track a manually-selected small patch of the subject's face, such as the nostril or the edge of the eyebrow. The optical flow tracker may lose the facial feature when the tracked image patch drifts away from the initially-selected feature or when a user makes a rapid head movement.

To address the problem of feature loss, the Kernel-Subset-Tracker was developed and incorporated into the Camera Mouse. The Kernel-Subset-Tracker uses the ***Kernel Semi-least Squares*** method and is an exemplar-based algorithm that uses a training set of representative images to produce online templates for positional tracking. Camera Mouse was augmented so that it can compute these templates in real time, employing kernel techniques traditionally used for classification. Chapter 6 details three versions of the Kernel-Subset-Tracker and their comparative performance to the optical-flow tracker under five different experimental conditions.

The experiments with test subjects show that augmenting the Camera Mouse with the Kernel-Subset-Tracker improves communication bandwidth statistically significantly. Tracking of facial features was accurate even during rapid head movements and extreme head orientations. Chapter 6 concludes by describing how the Camera Mouse augmented with the Kernel-Subset-Tracker enabled a stroke-victim with severe motion impairments to communicate via an on-screen keyboard.

Chapter 2

Conventions and Context

Let $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{N}$, Σ be the set of reals, rationals, natural numbers, and bits $\{0,1\}$. Let $X_{\geq 0}$ and $X_{>0}$ be the sets of non-negative and of positive elements of X. $[A] \stackrel{\text{def}}{=} 1$ if statement A holds, else $[A] \stackrel{\text{def}}{=} 0$. $\mathbf{1}_A(x) \stackrel{\text{def}}{=} [x \in A]$. The set of finite bit-strings is denoted by Σ^*. When it is clear from the context, we will use natural numbers and other finite objects interchangeably with their binary representations. The empty string is $\emptyset$. For $x \in \Sigma^*$, $(x0)^- = (x1)^- \stackrel{\text{def}}{=} x$. The set of all infinite binary sequences is denoted by Σ^∞. $\Sigma^{*\infty} \stackrel{\text{def}}{=} \Sigma^* \cup \Sigma^\infty$. For $x \in \Sigma^{*\infty}$, $y \in \Sigma^{*\infty}$, we say $x \sqsubseteq y$ iff $x = y$ or $x \in \Sigma^*$ and $y = xz$ for some $z \in \Sigma^{*\infty}$. The ith bit of $\alpha \in \Sigma^{*\infty}$ is denoted α_i, and its n bit prefix is denoted $\alpha_{\leq n}$. For $x \in \Sigma^*$, $\Gamma_x \subseteq \Sigma^\infty$ represents the set of all infinite strings $\alpha \in \Sigma^\infty$ where $x \sqsubseteq \alpha$. Thus Σ^∞ is a Cantor space and the set of intervals, $\{\Gamma_x : x \in \Sigma^*\}$, is a binary topological basis for Σ^∞. For compact sets Z of infinite strings, $Z_{\leq n} \stackrel{\text{def}}{=} \{\alpha_{\leq n} : \alpha \in Z\}$ and $Z_{<\infty} \stackrel{\text{def}}{=} \bigcup_{n\in\mathbb{N}} Z_{\leq n}$.

$\langle x \rangle \in \Sigma^*$ for $x \in \Sigma^*$ is the self-delimiting code that doubles every bit of x and changes the last bit of the result. The encoding of a finite set $\{x_n\}_{n=1}^m$ of strings is $\langle \{x_1, \ldots, x_m\} \rangle \stackrel{\text{def}}{=} \langle m \rangle \langle x_1 \rangle \ldots \langle x_m \rangle$, also denoted $\langle x_1, \ldots, x_m \rangle$. $\langle x, \alpha \rangle \stackrel{\text{def}}{=} \langle x \rangle \alpha$ for $x \in \Sigma^*$ and $\alpha \in \Sigma^\infty$. $\langle \alpha, \beta \rangle \stackrel{\text{def}}{=} \alpha_1\beta_1\alpha_2\beta_2\alpha_3\beta_3 \ldots$ for $\alpha, \beta \in \Sigma^\infty$. $D\Sigma^* \stackrel{\text{def}}{=} \{xy : x \in D, y \in \Sigma^*\}$. The bit length of a string x is $\|x\|$. $\|D\|$ is the number of elements of the set D, (not to be confused with $\|\langle D \rangle\|$). For $a \in \mathbb{R}_{>0}$, $\|a\| \stackrel{\text{def}}{=} \lceil |\log a| \rceil$.

For positive real functions f, by $\prec f$, $\succ f$, $\asymp f$, and $\lesssim f$, $\gtrsim f$, $\sim f$ we denote $\leq f + O(1)$, $\geq f - O(1)$, $= f \pm O(1)$ and $\leq f + O(\|f+1\|)$, $\geq f - O(\|f+1\|)$, $= f \pm O(\|f+1\|)$,

respectively. $\mathrm{Dom}(F)$ is the domain of F. $\langle F\rangle \stackrel{\text{def}}{=} \langle\{\langle x, F(x)\rangle\}\rangle$ if $x, F(x) \in \mathbb{N}$ and $\|\mathrm{Dom}(F)\| < \infty$. $\langle F\rangle \stackrel{\text{def}}{=} \langle i\rangle$ for functions F given by algorithms with Turing machine program i. $F(x) \stackrel{\text{def}}{=} \perp$ iff program i does not halt on input x. A predicate is a function with the range Σ and a domain of either $\mathbb{N}$ or a finite subset of $\mathbb{N}$. We say predicate λ extends predicate γ, $\lambda \supseteq \gamma$, iff $\mathrm{Dom}(\gamma) \subseteq \mathrm{Dom}(\lambda)$ and $\gamma(i) = \lambda(i)$ for $i \in \mathrm{Dom}(\gamma)$. $\gamma \cup \lambda$ denotes the union of predicates γ and λ consistent on the intersection of their domains.

Measures $P \in \mathscr{M}(X)$ on a locally compact space X are linear functionals on the space $C(X)$ of continuous functions $f : X \to \mathbb{R}$ with compact support (i.e. the closure of $X - f^{-1}(0)$). P is non-negative (i.e. $P \in \mathscr{M}(X)^+$) iff it is non-negative on $C(X)^+$. P is probabilistic (i.e. $P \in \mathscr{P}(X)$) iff it is normalized ($P(1) = 1$) and non-negative. P is extended to a larger class of functions in the standard way. $f : X \to \mathbb{R}_{\geq 0}$ is a P-test iff $P(f) \leq 1$. $P(D) \stackrel{\text{def}}{=} P(\mathbf{1}_D)$ for $D \subseteq X$. When $X = \Sigma^\infty$, $P(\Gamma_\emptyset) = 1$ and $P(\Gamma_x) = P(\Gamma_{x0}) + P(\Gamma_{x1})$ for all intervals Γ_x. We use shorthands $P(x)$ to mean $P(\Gamma_x)$ for $x \in \Sigma^*$ and $P(a)$ to denote $P(\{a\})$ for $a \in X = \mathbb{N}$. The uniform measure on Σ^∞ is denoted by $\mu(\Gamma_x) \stackrel{\text{def}}{=} 2^{-\|x\|}$. (Probabilistic) semi-measures σ are normalized ($\sigma(1) = 1 = -\sigma(-1)$) concave functionals on $C(X)$, non-negative on $C(X)^+$. They extend beyond $C(X)$ as is usual for internal measures. When $X = \Sigma^\infty$, $\sigma(\Gamma_x) \geq \sigma(\Gamma_{x0}) + \sigma(\Gamma_{x1})$ for all $x \in \Sigma^*$. We use shorthand $\sigma(x)$ to mean $\sigma(\Gamma_x)$.

$T_y(x)$ is the output of algorithm T (or $\perp$ if it does not halt) on input $x \in \Sigma^*$ and auxiliary input $y \in \Sigma^{*\infty}$. T is prefix-free if for all $x, s \in \Sigma^*$ with $s \neq \emptyset$, either $T_y(x) = \perp$ or $T_y(xs) = \perp$. We say $x \in \Sigma^*$ is total for T_y if there is a finite set $D \subset \Sigma^*$ where $\bigcup\{\Gamma_z : z \in D\} = \Omega$ and $T_y(xz) \neq \perp$ for all $z \in D$. We say T is left-total iff for all $x, s \in \Sigma^*$, $T_y(x1s) \neq \perp$ implies $x0$ is total for T_y. The complexity of $x \in \Sigma^*$ with respect to T_y is $\mathbf{K}_T(x|y) \stackrel{\text{def}}{=} \inf\{\|p\| : T_y(p) = x\}$.

There exist optimal for $\mathbf{K}$ prefix-free algorithms U, meaning that for all prefix-free

algorithms T, there exists $c_T \in \mathbb{N}$, where $\mathbf{K}_U(x|y) \leq \mathbf{K}_T(x|y) + c_T$ for all $x \in \Sigma^*$ and $y \in \Sigma^{*\infty}$. For example, one can take a universal prefix-free algorithm U, where for each prefix-free algorithm T, there exists $t \in \Sigma^*$, with $U_y(tx) = T_y(x)$ for all $x \in \Sigma^*$ and $y \in \Sigma^{*\infty}$. We can and will modify U to be left-total. $\mathbf{K}(x|y) \stackrel{\text{def}}{=} \mathbf{K}_U(x|y)$ is the Kolmogorov complexity of $x \in \Sigma^*$ relative to $y \in \Sigma^{*\infty}$. We say x is total relative to y iff it is total for U_y.

The halting sequence $\mathcal{H} \in \Sigma^\infty$ is the infinite string where $\mathcal{H}_i \stackrel{\text{def}}{=} [U(i) \neq \perp]$ for all $i \in \mathbb{N}$. A real function f is *r.e.* ($-f$ is co r.e.) if it is the supremum of an enumerable sequence of computable functions. The chain rule for Kolmogorov complexity is $\mathbf{K}(x, y) \asymp \mathbf{K}(x) + \mathbf{K}(y|\langle x, \mathbf{K}(x)\rangle)$. The universal probability of a finite set $D \subset \mathbb{N}$ is $\mathbf{m}(D|y) \stackrel{\text{def}}{=} \sum_{\{z\,:\,U_y(z)\in D\}} 2^{-\|z\|}$. We use shorthand $\mathbf{m}(x|y)$ to denote $\mathbf{m}(\{x\}|y)$ for $x \in \mathbb{N}$. $\mathbf{K}(x|y) \asymp \|\mathbf{m}(x|y)\|$. The mutual information in finite strings x and y relative to $z \in \Sigma^*$ is $\mathbf{I}(x : y \,|\, z) \stackrel{\text{def}}{=} \mathbf{K}(x|z) + \mathbf{K}(y|z) - \mathbf{K}(\langle x, y\rangle|z) \asymp \mathbf{K}(x|z) - \mathbf{K}(x|\langle y, \mathbf{K}(y), z\rangle)$.

We say $x \in \Sigma^*$ is to the left of $y \in \Sigma^*$, $x \trianglelefteq y$, iff $x \sqsupseteq y$ or there exists $u \in \Sigma^*$ with $u0 \sqsubseteq x$ and $u1 \sqsubseteq y$. $\mathbf{bb}(x|r) \stackrel{\text{def}}{=} \max_{y \trianglelefteq x} U_r(y)$ if x is total with respect to r, and $\mathbf{bb}(x|r) \stackrel{\text{def}}{=} \infty$, otherwise. $\mathbf{bb}(x) \stackrel{\text{def}}{=} \mathbf{bb}(x|\emptyset)$. $\mathbf{m}^p(a) \stackrel{\text{def}}{=} \sum_{\{z\,:\,z\trianglelefteq p,\, U(z)=a\}} 2^{-\|z\|}$ for total $p \in \Sigma^*$. The deficiency of randomness of $b \in \mathbb{N}$ with respect to computable measure P over $\mathbb{N}$ and to $v \in \mathbb{N}$ is $\mathbf{d}(b|P, v) \stackrel{\text{def}}{=} \|P(b)\| - \mathbf{K}(b|v)$. $\mathbf{d}(b|P) \stackrel{\text{def}}{=} \mathbf{d}(b|P, 0)$. The deficiency of randomness for $\alpha \in \Sigma^\infty$ with respect to computable measure P over Σ^∞ is $\mathbf{d}(\alpha|P) \stackrel{\text{def}}{=} \sup_n (\|P(\alpha_{\leq n})\| - \mathbf{K}(\alpha_{\leq n}))$.

The extension of information to include infinite sequences, with $a, b, \omega \in \Sigma^{*\infty}$ is $\mathbf{I}(a : b \,|\, \omega) \stackrel{\text{def}}{=} \log \sum_{x,y\in\Sigma^*} \mathbf{m}(x|\langle \omega, a\rangle)\mathbf{m}(y|\langle \omega, b\rangle)2^{\mathbf{I}(x:y|\omega)}$. Information follows conservation properties, with $\mathbf{I}(f(a) : b \,|\, \omega) \prec \mathbf{I}(a : b \,|\, \omega) + \mathbf{K}(\langle f\rangle|\omega)$ for computable function f. For $c \in \Sigma^*$, $\mathbf{I}(c; \mathcal{H}|\omega) \stackrel{\text{def}}{=} \mathbf{K}(c|\omega) - \mathbf{K}(c|\langle \omega, \mathcal{H}\rangle)$. The monotone complexity of a finite prefix-free set G of finite strings is $\mathbf{Km}(G) \stackrel{\text{def}}{=} \min\{\|p\| : U(p) \in G\Sigma^*,\, p \in \Sigma^*\}$. For $x \in \Sigma^*$, we use shorthand $\mathbf{Km}(x)$ to mean $\mathbf{Km}(\{x\})$. $\mathbf{M}(\Gamma_x)$ is the largest, up to a

constant multiplicative factor, r.e. semi-measure. We use shorthand $\mathbf{M}(x)$ to denote $\mathbf{M}(\Gamma_x)$. For finite prefix-free set $G \subset \Sigma^*$, $\mathbf{M}(G) \stackrel{\text{def}}{=} \sum_{x \in G} \mathbf{M}(x)$. Note that $\mathbf{M}(G)$ may differ from $\mathbf{M}\left(\cup_{x \in G} \Gamma_x\right)$. $\mathbf{KM}(G) \stackrel{\text{def}}{=} 1 - \lceil \log \mathbf{M}(G) \rceil$. We use shorthand $\mathbf{KM}(x)$ to mean $\mathbf{KM}(\{x\})$.

$\sqsubseteq$-sup is the supremum under the partial order of $\sqsubseteq$ on $\Sigma^{*\infty}$. A function $\nu : \Sigma^* \to \Sigma^*$ is prefix-monotone iff for all $p, q \in \Sigma^*$, $\nu(p) \sqsubseteq \nu(pq)$. Then $\overline{\nu} : \Sigma^{*\infty} \to \Sigma^{*\infty}$ denotes the unique extension of ν, where $\overline{\nu}(p) \stackrel{\text{def}}{=} \sqsubseteq\text{-sup}\left\{\nu(p_{\leq n}) : n \leq \|p\|, n \in \mathbb{N}\right\}$ for all $p \in \Sigma^{*\infty}$. For each lower-semicomputable (probabilistic) semi-measure σ over Σ^∞, there is a computable prefix-monotone function ν_σ such that for all $x \in \Sigma^*$, $\|\sigma(x)\| \asymp \|\mu\{\alpha : x \sqsubseteq \overline{\nu_\sigma}(\alpha),\ \alpha \in \Sigma^\infty\}\|$. Thus $\mathbf{KM}(x) \asymp \|\mu\{\alpha : x \sqsubseteq \overline{\nu_{\mathbf{M}}}(\alpha),\ \alpha \in \Sigma^\infty\}\|$.

Material about the history of algorithmic information theory can be found at [Li and Vitányi (2008)]. Information conservation laws were introduced and studied in [Levin (1974, 1984)]. Information asymmetry and the complexity of complexity were studied in [Gács (1975)]. A history of the origin of the mutual information of a string with the halting sequence can be found in [Vereshchagin and Vitányi (2004b)]. At a Tallinn conference in 1973, Kolmogorov formulated the notion of a two part code and introduced the structure function (see [Vereshchagin and Vitányi (2004b)] for more details). Aspects involving stochastic objects were studied in [Shen (1983, 1999); V'Yugin (1987, 1999)]. The work of Kolmogorov and the modelling of individual strings using a two-part code was expanded upon in [Vereshchagin and Vitányi (2004b); Gács et al. (2001)]. The development of algorithmic rate distortion theory can be seen in [Vereshchagin and Vitányi (2004a); Vereshchagin and Vitányi (2010)].

Chapter 3

Information and Games

The following chapter describes the work in [Epstein and Betke (2011a)]. Asymmetric information is defined to be $\mathbf{I}(x; y|z) \stackrel{\text{def}}{=} \mathbf{K}(x|z) - \mathbf{K}(x|y, z)$. The resource function $\mathbf{T} : \Sigma^* \to \mathbb{N}$ is the sum of the outputs of programs to U that are not to the right of program x with $\mathbf{T}(x) = U(x) + \sum_{u1 \sqsubseteq x,\, v} U(u0v)$. $\mathbf{T}(x)$ is undefined if $U(x)$ is undefined. Resource bounded complexity is defined by $\mathbf{K}_t(x) \stackrel{\text{def}}{=} \min\{p + \log t \, : \, U(p) = x, \, \mathbf{T}(x) = t\}$.

We define players A and B as two sets containing strings of size n. Each string x in the intersection set $A \cap B$ represents a particular "interaction" between players A and B. In this chapter, we will use the terms *string* and *interaction* interchangeably. This set representation can be used to encode games and instances of the cybernetic agent model. Uncertainties in instances of both domains can be encoded into the size of the intersections. The amount of uncertainty between players is equal to $|A \cap B|$. If the interaction between the players is deterministic then $|A \cap B| = 1$. If uncertainty exists, then multiple interactions are possible and $|A \cap B| > 1$. We say that player A *interacts* with B if $|A \cap B| > 0$.

3.1 Games

Sets can be used to encode adversaries in sequential games [Russell and Norvig (2009)], where agents exchange a series of actions over a finite number of plies. Each game or interaction consists of the recording of actions by adversaries **A** and **B**, with

$x = (a_1, b_1)(a_2, b_2)(a_3, b_3)$ for a game of three rounds. The player (set) representation A of adversary **A** is the set of games representing all possible actions by **A**'s adversary with **A**'s responding actions, and similarly for player B representing adversary **B**. An example game is rock-paper-scissors where adversaries **A** and **B** play two sequential rounds with an action space of $\{R, P, S\}$. Adversary **A** only plays rock, whereas adversary **B** first plays paper, then copies his adversary's play of the first round. The corresponding players (sets) A and B can be seen in Fig. 3·1. The intersection set of A and B contains the single interaction $x =$"$(R, P)(R, R)$," which is the only possible game (interaction) that **A** and **B** can play.

Example 1 (Chess Game)**.** *We use the example of a chess game with uncertainty between two players: Anatoly as white and Boris as black. An interaction $x \in A \cap B$ between Anatoly and Boris is a game of chess played for at most m plies for each player, with $x = a_1 b_1 a_2 b_2 \dots a_m b_m = a b_{1:m}$. The chess move space $\mathcal{V} \subset \{0,1\}^*$ has a short binary encoding, whose precise definition is not important. If the game has not ended after m rounds, then the game is considered a draw. Both players are nondeterministic, where at every ply, they can choose from a selection of moves. Anatoly's decisions can be represented by a function $f_A : \mathcal{V}^* \to 2^{\mathcal{V}}$ and similarly Boris' decisions by f_B. Anatoly can be represented by a set A, with $A = \{a b_{1:m} : \forall_{1 \le k \le m}\ a_k \in f_A(a b_{1:k-1})\}$, and similarly Boris by set B. Their intersection, $A \cap B$, represents the set of possible games that Anatoly and Boris can play together.*

Generally, sets can encode adversaries of games, with their interactions consisting of equilibriums [Russell and Norvig (2009)]. A game is defined as (p, q) with the adversaries represented by normalized payoff functions p and q of the form $\{0,1\}^n \times \{0,1\}^n \to [0,1]$. The set of equilibriums is $\{\langle x, y\rangle \ : \ p(x,y) = q(y,x) = 1\}$. For each payoff function p there is a player $A = \{\langle x, y\rangle \ | \ p(x,y) = 1\}$, and for each payoff function q there is player B. The intersection of A and B is equal to the set of equilibriums of p and q.

A	B
(R,R)(R,R)	**(R,P)(R,R)**
(R,R)(R,P)	(R,P)(P,R)
(R,R)(R,S)	(R,P)(S,R)
(R,P)(R,R)	(P,P)(R,P)
(R,P)(R,P)	(P,P)(P,P)
(R,P)(R,S)	(P,P)(S,P)
(R,S)(R,R)	(S,P)(R,S)
(R,S)(R,P)	(S,P)(P,S)
(R,S)(R,S)	(S,P)(S,S)

Figure 3·1: The set representation of players A and B playing two games of rock, paper, scissors. The intersection set of A and B contains the single interaction $x =$"$(R,P)(R,R)$."

3.2 Cybernetic Agent Model

The interpretation of "interaction as intersection" is also applicable to the cybernetic agent model used in Universal Artificial Intelligence [Hutter (2004)]. With the cybernetic agent model, there is an agent and an environment communicating in a series of cycles $k = 1, 2, \ldots$. At cycle k, the agent performs an action $y_k \in \mathcal{Y}$, dependent on the previous history $yx_{<k} = y_1 x_1 \ldots y_{k-1} x_{k-1}$. The environment accepts the action and in turn outputs $x_k \in \mathcal{X}$, which can be interpreted as the kth perception of the agent, followed by cycle $k+1$ and so on. An agent is defined by a deterministic policy function $p : \mathcal{X}^* \to \mathcal{Y}^*$ with $p(x_{<k}) = y_{1:k}$ to denote output $y_{1:k} = y_1 y_2 \ldots y_k$ on input $x_{<k} = x_1 x_2 \ldots x_{k-1}$. We use the terms *policy* and *agent* interchangeably. The inputs are separated into two parts, $x_k \equiv r_k o_k$, with $r_k = r(x_k)$ representing the reward and o_k representing the observation. We say $r(x_{1:m}) = \sum_{i=1}^{m} r(x_i)$ and we assume bounds on rewards with $0 \leq r_k \leq c$ for all k. There is uncertainty in the environment; it can be represented by a probability distribution over infinite strings, where $\Phi(x_1 \ldots x_n)$ is the probability that an infinite string starts with $x_1 \ldots x_n$. In Hutter's notation [Hutter (2004)], an underlined argument $\underline{x}_k$ is a probability variable and non-underlined arguments x_k represent the condition with $\Phi(x_{<n}\underline{x}_n) = \Phi(\underline{x}_{1:n})/\Phi(\underline{x}_{<n})$.

The probability that the environment reacts with $x_1 \dots x_n$ under agent output $y_1 \dots y_n$ is $\Phi(y_1 \underline{x}_1 \dots y_n \underline{x}_n) = \Phi(y \underline{x}_n)$. The environment is chronological, in that input x_i only depends on $yx_{<i}y_i$. The horizon m of the interaction is the number of cycles of the interaction. The *value* of agent p in environment Φ is the expected reward sum $V_{1:m}^{p,\Phi} = \sum_{x_{1:m}} r(x_{1:m}) \Phi(y \underline{x}_{1:m})_{|y_{1:m}=p(x_{<m})}$. The optimal agent that maximizes value $V_{1:m}^{p,\Phi}$ is $p^\Phi = \arg\max_p V_{1:m}^{p,\Phi}$, with value $V_{1:m}^{*,\Phi} = V_{1:m}^{p^\Phi,\Phi}$. The optimal expected reward given a partial history $yx_{1:k}$ is $V_{1:m}^{p,\Phi}(yx_{1:k})$.

It is possible to construct players (sets) A and B from the agent p and environment Φ where A "interacts" (intersects) with B only if agent p can achieve a certain level of reward in Φ.To translate p and Φ to A and B, we fix two parameters: a time horizon m and a difficulty threshold d. For every agent p, there is a player A_m^p, with $A_m^p = \{yx_{1:m} \ : \ y_{1:m} = p(x_{<m})\}$. There are several possible ways to construct a set B from an environment Φ. One direct method is for every environment Φ, to define a player $B_{m,d}^\Phi$, with $B_{m,d}^\Phi = \{yx_{1:m} \ : \ r(x_{1:m}) \geq d, \ \Phi(y \underline{x}_{1:m}) > 0\}$. Player $B_{m,d}^\Phi$ represents all possible histories of Φ (however unlikely) where the reward is at least d. If $A_m^p \cap B_{m,d}^\Phi = \emptyset$, then environment Φ is "too difficult" for the agent p; there is no interaction where the agent can receive a reward of at least d. We say the agent p *interacts* with the environment Φ at time horizon m and difficulty d if $A_m^p \cap B_{m,d}^\Phi \neq \emptyset$.

Example 2 (Peter and Magnus)**.** *We present a cybernetic agent model interpretation of chess with reward based players Peter and Magnus (same rules as example 1). Peter, the agent p, has to be deterministic whereas Magnus, the environment Φ, has uncertainty. At cycle k, each action y_k is Peter's move and each perception x_k is Magnus' move. At ply m in the chess game, Magnus returns a reward of 1 if Peter has won. In rounds where the game is unfinished or if Peter loses or draws, the reward is 0. The player (set) A_m^p represents Peter's plays for m rounds. The player (set) for Magnus with difficulty threshold $d = 1$ and m plies, $B_{m,1}^\Phi$ is the set of all games that Magnus loses in m rounds or less. If $A_m \cap B_{m,1}^\Phi = \emptyset$, then Peter cannot* interact *with Magnus at difficulty level* 1*; Peter can never beat Magnus at chess in m rounds or less. If $A_m \cap B_{m,1}^\Phi \neq \emptyset$ then Peter can beat Magnus at a game of chess in m rounds*

or less.

Another construction of a player $D^{\Phi}_{m,d}$ with respect to environment Φ, is $D^{\Phi}_{m,d} = \{yx_{1:m} \ : \ \forall_k \ \ V^{*,\Phi}_{1:m}(yx_{1:k})/V^{*,\Phi}_{1:m} \geq d\}$.With this interpretation, player $D^{\Phi}_{m,d}$ represents all histories where at each time k, $1 \leq k \leq m$, an agent can potentially achieve an expected reward of at least d times the optimal expected reward. If $A^p_m \cap D^{\Phi}_{m,d} = \emptyset$, then environment Φ is "too difficult" for the agent p; there is no interaction where at *every* cycle k the agent has the potential to receive an expected reward of at least $dV^{*,\Phi}_{1:m}$.

3.3 Player Strategy Learning

Players A and B can learn information about each other's strategies from a single interaction (game) $x \in A \cap B$ or from their entire interaction set (all possible games) $A \cap B$. The *capacity* of a player A is the maximum amount of information that A can receive about another player through all possible interactions, i.e. their interaction set. It is equal to the log of the number of possible subsets that it can have, $\log 2^{|A|} = |A|$. We define the deficiency of randomness of a subset S with respect to A to be $\delta(S|A) = |A| - \mathbf{K}(S|A)$, for $S \subseteq A$ and ∞ otherwise.

Example 3 (Capacity)**.** *Boris B uses a range of black openings whereas Bill B' uses only the Sicilian defence. So Boris has a higher capacity, $|B| \gg |B'|$, and can potentially learn more than Bill.*

Example 4 (Randomness Deficiency)**.** *Let A be the chess games played by Anatoly. Bob is a simple player B', who only moves his knight back and forth. Set $S = A \cap B'$ represents all A's games with B'. The randomness deficiency of these games, $\delta(S|A)$, is high, as S is easily computable from A, with $\mathbf{K}(S|A) \ll |A|$. Let $T \subseteq A$, in which $T = A \cap B$ are games played between Anatoly and Boris, who uses a range of chess strategies unknown to Anatoly. Then $\delta(T|A)$ is low and $\mathbf{K}(T|A)$ is high.*

If A views every interaction in $A \cap B$, the amount of information B reveals about itself is, $\mathbf{I}(A \cap B; B|A)$, the mutual information between B and $A \cap B$, given A.

This term can be reduced to $\mathbf{K}(A \cap B|A) - \mathbf{K}(A \cap B|A, B) \asymp \mathbf{K}(A \cap B|A)$. We define the amount of knowledge that A received about B from the interaction set as: $\mathbf{R}(B|A) = \mathbf{K}(A \cap B|A)$.

The higher the randomness deficiency, $\delta(A \cap B|A)$, of an interaction set, $A \cap B$, with respect to player A, the less information, $\mathbf{R}(B|A)$, player A can receive about its opponent B, with $\mathbf{R}(B|A)+\delta(A\cap B|A) = |A|$. Player A receives the most information about its opponent when the randomness deficiency is $\delta(A \cap B|A) = 0$.

Example 5. *Let Anatoly, A, and Bob, B', be the players of example 4. Bob has a simple strategy and has a lower capacity $|B'| \ll |A|$, but he learns a lot from Anatoly, with $\delta(A \cap B'|B') \approx 0$ and $\mathbf{R}(A|B') \approx |B'|$. Anatoly learns very little from Bob, with $\mathbf{R}(B'|A) \approx 0$ and $\delta(A \cap B'|A) \approx |A|$.*

3.4 Single Interaction Learning

Players can reveal information about themselves through a single interaction. The amount of information that A received about B from their interaction x is $\mathbf{I}(x; B|A) = \mathbf{K}(x|A) - \mathbf{K}(x|A, B)$. We define the deficiency of randomness of an interaction x with respect to both players to be $\delta(x|A, B) = \log |A \cap B| - \mathbf{K}(x|A, B)$ for $x \in A \cap B$ and ∞ otherwise. If $\delta(x|A, B)$ is small, then x represents a typical interaction. The information passed from player B to player A through a single interaction is represented by $\mathbf{I}(x; B|A) + \delta(x|A) = \log |A|/|A \cap B| + \delta(x|A, B)$. The information passed between players through a single interaction with the same capacity is $\mathbf{I}(x; B|A) + \delta(x|A) = \mathbf{I}(x; A|B) + \delta(x|B) + O(1)$.

Example 6. *Anatoly A plays a game x with Boris B who has the same capacity with $|A| = |B|$. Anatoly tricks Boris with a King's gambit and the game x follows a series of moves extremely familiar to Anatoly. Boris reacts with the most obvious move at every turn. In this case the game is simple to Anatoly, with $\delta(x|A)$ being large and $\mathbf{I}(x; B|A)$ being small. The game is new to Boris with $\delta(x|B)$ being small and $\mathbf{I}(x; A|B)$ being large. Thus Boris learns more than Anatoly from x.*

If the players have a deterministic interaction, then $A \cap B = \{y\}$ and the information A received from B reduces to $\mathbf{I}(y; B|A) + \delta(y|A) = \log |A|$.

3.5 Player Approximation and Interaction Complexity

The following theorem shows that if a string x is contained by a large number of sets of a certain complexity, then it is contained by a simpler set [Vereshchagin and Vitányi (2004a)]. Let $\mathcal{F} \in \Sigma^*$ be a finite set of sets of strings $\mathcal{F} = \{F_n\}_{n=1}^m$.

Theorem 1 (Vereshchagin and Vitányi (2010))**.** *Let $\mathcal{F}$ be a family of subsets of a set of strings $\mathcal{G}$. If $x \in \mathcal{G}$ is an member of each of 2^k sets $F \in \mathcal{F}$ with $\mathbf{K}(F) \leq r$, then x is a member of a set F' in $\mathcal{F}$ with $\mathbf{K}(F') \leq r - k + O(\log k + \log r + \log\log |\mathcal{G}| + \mathbf{Km}(\mathcal{F}))$.*

Given are players A and B who *interact*, in that $A \cap B \neq \emptyset$. We show that there exists an interacting player B' that has complexity bounded by the mutual information of A and B. The following theorem applies theorem 1 in the following way. X is a set of sets such that if $G \in X$, then $\mathbf{K}_t(G) \leq h$ for some particular upper bound h.

Theorem 2. *Given are a player space $\mathcal{B}$ and players A and $B \in \mathcal{B}$ with $A \cap B \neq \emptyset$. Then there exists a player $B' \in \mathcal{B}$ with $A \cap B' \neq \emptyset$, and $\mathbf{K}(B') \leq \mathbf{I}(A; B) + O(s)$, with $s = \log \mathbf{K}(B) + \log \mathbf{K}_t(A) + \mathbf{K}(\mathcal{B})$.*

Proof. Let $r = \mathbf{K}(B)$, $h = \mathbf{K}_t(A)$, and $q = 2^{\mathbf{K}(\mathcal{B})}$. We define $\mathcal{G} = \{\langle S\rangle \ : \ \mathbf{K}_t(S) \leq r\}$, with $\langle S\rangle$ being an encoding of set S. This implies $\log\log |\mathcal{G}| = O(\log h)$. We define $\mathcal{F}$ with a recursive function $\lambda : \mathcal{B} \to \mathcal{F}$, with $\lambda(S) = \{\langle T\rangle \mid \mathbf{K}_t(T) \leq h, S \cap T \neq \emptyset\}$. It must be $\mathbf{K}(\lambda) = O(\log h)$. The $\mathbf{Km}$ complexity of $\mathcal{F}$ requires the encoding of $\mathcal{B}$ and λ, and so $\mathbf{Km}(\mathcal{F}) = O(\log hq)$. Thus if $\langle T\rangle \in \lambda(S)$, then $T \cap S \neq \emptyset$. Let N be the number of sets $S \in \mathcal{B}$, with $\mathbf{K}(S) \leq r$ and $S \cap A \neq \emptyset$. Thus $\mathbf{K}(B|A) \leq \log N + O(\log hqr)$, as there is a program, when given A, r, $\mathcal{B}$, and an index of size $\lceil \log N \rceil$, that can return any such S. By the application of Theorem 1, with $x = \langle A\rangle$ and $k = \lfloor \log N \rfloor$, there is a set $F \in \mathcal{F}$ with $x \in F$ and

$\mathbf{K}(F) \leq r - k + O(\log hqr) \leq \mathbf{K}(B) - \mathbf{K}(B|A) + O(\log hqr) = \mathbf{I}(A;B) + O(\log hqr)$. A set $B' \in \mathcal{B}$, with $\lambda(B') = F$, can be easily recovered from F by enumerating all sets in $\mathcal{B}$, applying λ to each one, and selecting the first one which produces F. So $\mathbf{K}(B') \leq \mathbf{K}(F) + O(\log q) \leq \mathbf{I}(A;B) + O(\log hqr)$. Since $\langle A \rangle \in \lambda(B')$, it must be that $A \cap B' \neq \emptyset$. □

However the bound of $\log \mathbf{K}_t(\cdot)$ can be improved upon, and be replaced with mutual information with the halting sequence $\mathbf{I}(\cdot : \mathcal{H})$. This initial theorem led to the work in [Epstein and Levin (2011)] which is expanded upon in the next chapter.

Chapter 4

Sets Have Simple Members

The results in this chapter were produced in collaboration with Leonid A. Levin. Many intellectual and computing tasks require guessing the hidden part of the environment from available observations. In different fields these tasks have various names, such as Inductive Inference, Extrapolation, Passive Learning, etc. The relevant part of the environment includes not only what interests the guesser, but also all data that might influence the guess; it can be represented as an, often huge, string $x \in \{0,1\}^*$. The known observations restrict it to a set $D \ni x$. D is typically enormous, and many situations allow replacing it with a much more concise theory representing the relevant part of what is known about x. Yet, such approaches are *ad hoc* and secondary: raw observations are the ultimate source.

One popular approach to guessing, the "Occam Razor," tells to focus on the simplest members of D. (In words, attributed to A. Einstein, "A conjecture should be made as simple as it can be, but no simpler.") Its implementations vary: if two objects are close in simplicity, there may be legitimate disagreements of which is slightly simpler. This ambiguity is reflected in formalization of "simplicity" via the Kolmogorov Complexity function $\mathbf{K}(x)$ - the length of the shortest prefix program generating x: $\mathbf{K}$ is defined only up to an additive constant depending on the programming language. This constant is small compared to the usually huge whole bit-length of x. More mysterious is the justification of this Occam Razor principle.

A more revealing philosophy is based on the idea of "Prior". It assumes the

guessing of $x \in D$ is done by restricting to D an *a priori* probability distribution on $\{0,1\}^*$. Again, subjective differences are reflected in ignoring moderate factors: say in asymptotic terms, priors different by a $\theta(1)$ factors are treated as equivalent. The less we know about x (before observations restricting x to D) the more "spread" is the prior, i.e. the smaller would be the variety of sets that can be ignored due to their negligible probability. This means that distributions truly prior to any knowledge, would be the largest up to $\theta(1)$ factors. Among enumerable (i.e. generated by randomized algorithms on their outputs) distributions, such largest prior does in fact exist and is $\mathbf{m}(x) \asymp 2^{-\mathbf{K}(x)}$.

These ideas are developed in [Solomonoff (1964)] and many subsequent papers. Yet, they immediately yield a reservation: the simplest objects have **each** the highest universal probability, but it may still be negligible compared to the **combined** probability of complicated objects in D. This suggests that the general inference situation may be much more obscure than the widely believed Occam Razor principle describes it.

The work in [Epstein and Levin (2011)] and this chapter show this could not happen, except as a purely mathematical construction. Any such D has high information $\mathbf{I}(D : \mathcal{H})$ about the Halting Problem $\mathcal{H}$ ("Turing's Password"). So, there are no ways to generate such D: see this informational version of Church-Turing Thesis discussed in the appendix of [Levin (2002)]. We call ***exotic*** those numbers (and any object they encode) which have high mutual information with $\mathcal{H}$.

Consider finite sets D containing only strings of high ($\gtrsim k$) complexity. One way to find such D is to generate at random a small number of strings $x \in \{0,1\}^k$. With a little luck, all x would have high complexity, but D would contain virtually all information about each of them.

Another (less realistic) method is to gain access to the halting problem sequence

$\mathcal{H}$ and use it to select for D strings x of complexity $\sim k$ from among all k-bit strings. Then D contains little information about most its members but much information about $\mathcal{H}$.

Another way is to combine both methods. Let v_h be the set of all strings vs with $\mathbf{K}(vs) \sim \|vs\|=\|v\|+h$. Then $\mathbf{K}(x) \sim i+h$, $\mathbf{I}(D:x) \sim i$, and $\mathbf{I}(D;\mathcal{H}) \sim h$ for most i-bit v and $x \in D = v_h$. We show no D can be better: they all contain strings of complexity $\lesssim \min_{x\in D} \mathbf{I}(D:x)+\mathbf{I}(D;\mathcal{H})$.

The following lemma shows every *non-exotic* natural number b is the typical member of a simple probability space. For this chapter, we use the following slice $\Lambda(b|y) \stackrel{\text{def}}{=} \min_{\{U_y(v)=Q\in\mathscr{P}(\mathbb{N}),\|\text{supp}(Q)\|<\infty\}} \|v\|+3\|\mathbf{d}(b|Q,\langle v,y\rangle)\|$ of Kolmogorov's structure function.

Lemma 1 (Epstein and Levin (2011))**.** *For $a \in \mathbb{N}$, $\Lambda(a) \lesssim \mathbf{I}(a;\mathcal{H})$.*

Proof. Let $U(uw) = a$, with $\|uw\| = \mathbf{K}(a)$, u be total and u^- be not. Let $Q \in \mathscr{P}(\mathbb{N})$ with $Q(b) \stackrel{\text{def}}{=} \sum_{p:U(up)=b} 2^{-\|p\|}$. Thus $\|Q(a)\| \asymp \|w\|$ and $\|\text{supp}(Q)\| < \infty$ by compactness arguments. Let $U(v) = Q$, with $\|v\| = \mathbf{K}(Q) \prec \|u\| + \mathbf{K}(\|u\|)$. Furthermore $\|u\| + \|w\| = \mathbf{K}(a) \prec \mathbf{K}(a|v) + \mathbf{K}(v) \prec \mathbf{K}(a|v) + \|u\| + \mathbf{K}(\|u\|)$ implies $\|w\| - \mathbf{K}(\|u\|) \prec \mathbf{K}(a|v)$. Thus $\mathbf{d}(a|Q,v) = \|Q(a)\| - \mathbf{K}(a|v) \prec \mathbf{K}(\|u\|)$. Thus $\Lambda(a) \leq \mathbf{K}(Q) + 3\|\mathbf{d}(a|Q,v)\| \leq \|u\| + \mathbf{K}(\|u\|) + O(\|\mathbf{K}(\|u\|)\|) \lesssim \|u\|$. Since U is left total, u can be computed from $\mathcal{H}$ and $\|u\|$. So $\mathbf{K}(u|\mathcal{H}) \prec \mathbf{K}(\|u\|)$. Thus $\mathbf{K}(a|\mathcal{H}) \prec \mathbf{K}(\|u\|)+\|w\|$ implies $\mathbf{I}(a;\mathcal{H}) = \mathbf{K}(a)-\mathbf{K}(a|\mathcal{H}) \succ \|u\|-\mathbf{K}(\|u\|) \gtrsim \Lambda(a)$.□

Lemma 2. *Let computable W:$\mathbb{N}\to\mathbb{R}_{>0}$, η:$\mathbb{N}\to\mathbb{R}_{>0}$ have $\sum_{a\in\mathbb{N}} W(a)\eta(a)\le 1$. If finite set $D\subset\mathbb{N}$ has $\sum_{a\in D}\eta(a)\ge 1$, then $\mathbf{K}(a)\prec -\log W(a)+\Lambda(D)$ for some $a\in D$.*

Proof. Let v, $Q\stackrel{\text{def}}{=}U(v)$ minimize $\Lambda(D)$ and $d\stackrel{\text{def}}{=}|\mathbf{d}(D|Q,v)|$. We condition Q on the largest set $\mathcal{X}\subseteq\text{supp}(Q)$ containing solely $R\subset\mathbb{N}$ with $\sum_{a\in R}\eta(a)\ge 1$ and $\max_{a\in R} 2\eta(a)(c+d)\le 1$, for some constant c to be determined later. $D\in\mathcal{X}$, otherwise there exists $a\in D$ with $-\log\eta(a)\prec\log d$. $\sum_{a\in\mathbb{N}} W(a)(\eta)\le 1$ implies $\mathbf{K}(a)\prec -\log\left(W(a)\eta(a)\right)\prec -\log\left(W(a)/d\right)\asymp -\log W(a)+\|d\|\prec -\log W(a)+3\|d\|+\|v\|=-\log W(a)+\Lambda(D)$.

$S\stackrel{\text{def}}{=}\bigcup\mathcal{X}$. Let $t_G:\mathcal{X}\to\mathbb{R}_{\ge 0}$, where for each $G\subseteq S$, $t_G(R)\stackrel{\text{def}}{=}0$ if $G\cap R\ne\emptyset$ and $t_G(R)\stackrel{\text{def}}{=}e^{c+d}$, otherwise. Let δ be a random subset of S, with elements a chosen independently with probability $2\eta(a)(c+d)$.

So we have $\mathbf{E}_\delta[Q(t_\delta)]=\sum_{R\in\mathcal{X}} Q(R)\mathbf{E}_\delta[t_\delta(R)]=\sum_{R\in\mathcal{X}} Q(R)\Pr\left(\delta\cap R=\emptyset\right)e^{c+d}$ $=\sum_{R\in\mathcal{X}} Q(R)\prod_{a\in R}(1-2\eta(a)(c+d))e^{c+d}$. Using $(1-m)e^m\le 1$ for $m\in[0,1]$ we get the inequality $\mathbf{E}_\delta[Q(t_\delta)]\le\sum_{R\in\mathcal{X}} Q(R)\prod_{a\in R} e^{-2\eta(a)(c+d)}e^{c+d}=\sum_{R\in\mathcal{X}} Q(R)\exp\{(1-2\sum_{a\in R}\eta(a))(c+d)\}\le\sum_{R\in\mathcal{X}} Q(R)e^{-(c+d)}\le\sum_{R\in\mathcal{X}} Q(R)/e<0.5$. And $\mathbf{E}_\delta[Q(t_\delta)]<0.5$ implies $\Pr(Q(t_\delta)\le 1)>0.5$. We use measures $P_G\in\mathscr{M}(\mathbb{N})^+$, indexed by $G\subseteq S$, where $P_G(a)\stackrel{\text{def}}{=}[a\in G]W(a)/4(c+d)$.

So this gives us $\mathbf{E}_\delta[P_\delta(\mathbb{N})]=\sum_{a\in S}\Pr(a\in\delta)W(a)/4(c+d)=\sum_{a\in S}0.5W(a)\eta(a)\le 0.5$. So $\Pr(P_\delta(\mathbb{N})\le 1)\ge 0.5$, and since $\Pr(Q(t_\delta)\le 1)>0.5$, there exists $G\subseteq S$ with $Q(t_G)\le 1$ and $P_G(\mathbb{N})\le 1$. Such G can be found with brute force search given c, d, and v, so $\mathbf{K}(G|\langle c,d,v\rangle)=O(1)$.

For each $a\in G$, $\mathbf{K}(a|\langle d,G\rangle)\prec -\log P_G(a)\asymp -\log W(a)+\|d\|$. It must be that $G\cap D\ne\emptyset$. Assume it is not. Then $t_G(D)=e^{d+c}$; Since t_G is a Q-test, $Q(\{R: t_G(R)=e^{d+c}\})\le e^{-d-c}$. Thus every R with $t_G(R)=e^{d+c}$ can be identified using c, d, G, v, and a prefix-free code of size $\asymp\|Q(R)\|-\|e^{d+c}\|$. Therefore $\mathbf{K}(D|\langle c,d,G,v\rangle)\prec\|Q(D)\|-(\lg e)(c+d)$. Thus, for proper choice of c solely dependent on U, $d=|\mathbf{d}(D|Q,v)|\ge$

$\|Q(D)\|-\mathbf{K}(D|v) \geq \|Q(D)\|-\mathbf{K}(D\,|\langle c,d,G,v\rangle)-\mathbf{K}(G|\langle c,d,v\rangle)-\mathbf{K}(\langle c,d\rangle|v)-O(1) \geq (\lg e)(c+d)-\mathbf{K}(c)-\mathbf{K}(d)-O(1) > d$. This contradiction proves $G\cap D\neq\emptyset$.

So there exists $a\in D\cap G$ with $\mathbf{K}(a) \prec \mathbf{K}(a|\langle d,G\rangle)+\mathbf{K}(G|\langle d,v\rangle)+\mathbf{K}(\langle d,v\rangle) \prec -\log W(a)+\|d\|+\mathbf{K}(G|\langle d,v\rangle)+\mathbf{K}(\langle d,v\rangle) \prec -\log W(a)+\|d\|+\mathbf{K}(\langle d,v\rangle) \prec -\log W(a)+\|d\|+\mathbf{K}(d)+\mathbf{K}(v) \prec -\log W(a)+\Lambda(D)$. □

Proposition 1. *If $b\in\Sigma^*$ is total and b^- is not, and $x\in\Sigma^*$, then $\|b\| \prec \mathbf{K}(b)$ and $\mathbf{K}(b)+\mathbf{I}(x;\mathcal{H}|b) \lesssim \mathbf{I}(x\,;\mathcal{H})+\mathbf{K}(b|\langle x,\|b\|\rangle)$.*

As U is left total, b is computable from $\|b\|$ and $\mathcal{H}$. So $\mathbf{K}(b|\mathcal{H}) \prec \mathbf{K}(\|b\|)$. The list D of integers of complexity $<\|b\|$ is computable from b. So $\mathbf{K}(n|b)=O(1)$ for $n\stackrel{\rm def}{=}\min\{\mathbb{N}\backslash D\}$ implies $\|b\|\leq\mathbf{K}(n)\prec\mathbf{K}(n,b)\prec\mathbf{K}(n|b)+\mathbf{K}(b)\asymp\mathbf{K}(b)$.

The chain rule $\mathbf{K}(b)+\mathbf{K}(x|b,\mathbf{K}(b))\asymp\mathbf{K}(x)+\mathbf{K}(b|x,\mathbf{K}(x))$ implies $\mathbf{K}(b)+\mathbf{K}(x|b)\prec\mathbf{K}(x)+\mathbf{K}(b|x)+\mathbf{K}(\mathbf{K}(b))$. Now $\mathbf{K}(b)+\mathbf{K}(x|b)-\mathbf{K}(x|\langle b,\mathcal{H}\rangle)\prec\mathbf{K}(x)+\mathbf{K}(b|x)+\mathbf{K}(\mathbf{K}(b))-\mathbf{K}(x|\langle b,\mathcal{H}\rangle)\prec\mathbf{K}(x)+\mathbf{K}(b|x)+\mathbf{K}(\mathbf{K}(b))-\mathbf{K}(x|\mathcal{H})+\mathbf{K}(b|\mathcal{H})=\mathbf{I}(x;\mathcal{H})+\mathbf{K}(b|x)+\mathbf{K}(\mathbf{K}(b))+\mathbf{K}(b|\mathcal{H})\leq\mathbf{I}(x;\mathcal{H})+\mathbf{K}(b|x)+O(\log\|b\|)\leq\mathbf{I}(x;\mathcal{H})+\mathbf{K}(b|\langle x,\|b\|\rangle)+\mathbf{K}(\|b\|)+O(\log\|b\|)$. So $\mathbf{K}(b)+\mathbf{I}(x;\mathcal{H}|b)\lesssim\mathbf{I}(x;\mathcal{H})+\mathbf{K}(b|\langle x,\|b\|\rangle)$. □

Theorem 3. *Given r.e. $F:\mathbb{N}\to\mathbb{N}_{>0}$, for all finite sets $D\subset\mathbb{N}$, $s\stackrel{def}{=}\lceil\log\sum_{a\in D}F(a)\mathbf{m}(a)\rceil-1\leq\log\max_{a\in D}(F(a)\mathbf{m}(a))+\mathbf{I}(D;\mathcal{H})+O(\mathbf{K}(s)+\|\mathbf{I}(D;\mathcal{H})\|)$.*

Proof. Let F be the supremum of an enumerable sequence F_i of non-negative computable functions. Let b be the shortest total string with $\sum_{a\in D}F_N(a)\mathbf{m}^b(a)>2^s$, where $N\stackrel{\rm def}{=}\mathbf{bb}(b)$. Let $h_x\stackrel{\rm def}{=}\mathbf{I}(D;\mathcal{H}|x)$.

Let $W(a)\stackrel{\rm def}{=}2^s/F_N(a)$ and $\eta(a)\stackrel{\rm def}{=}2^{-s}F_N(a)\mathbf{m}^b(a)$, where $\mathbf{K}(\langle W,\eta\rangle|\langle b,s\rangle)=O(1)$. Lemma 2 relativized to $\langle b,s\rangle$, gives $a\in D$ with $\mathbf{K}(a|\langle b,s\rangle)\prec-\log W(a)+\Lambda(D|\langle b,s\rangle)=\log F_N(a)-s+\Lambda(D|\langle b,s\rangle)$.

Due to lemma 1, $\mathbf{K}(a|\langle b,s\rangle)\leq\log F(a)-s+h_{\langle b,s\rangle}+O(\log h_{\langle b,s\rangle})$. So $s\leq\log F(a)-\mathbf{K}(a)+\mathbf{K}(\langle b,s\rangle)+h_{\langle b,s\rangle}+O(\log h_{\langle b,s\rangle})\leq\log(F(a)\mathbf{m}(a))+\mathbf{K}(b)+h_b+O(\mathbf{K}(s)+\log h_b)$. By proposition 1, $\mathbf{K}(b)+h_b\lesssim h_\emptyset+\mathbf{K}(b|\langle D,\|b\|\rangle)$. Therefore $s\leq\log(F(a)\mathbf{m}(a))+$

$h_\emptyset + \mathbf{K}(b|\langle D, \|b\|\rangle) + O(\mathbf{K}(s) + \log(h_\emptyset + \mathbf{K}(b|\langle D, \|b\|\rangle)))$. $\mathbf{K}(b|\langle D, \|b\|\rangle) \prec \mathbf{K}(s)$. So $s \leq \log F(a)\mathbf{m}(a) + h_\emptyset + O(\mathbf{K}(s) + \log h_\emptyset)$. □

Corollary 1 (Epstein and Levin (2011))**.** *For any finite set* $D \subset \mathbb{N}$,
$\min_{a\in D} \mathbf{K}(a) \leq \|\mathbf{m}(D)\| + \mathbf{I}(D\,;\mathcal{H}) + O(\mathbf{K}(\|\mathbf{m}(D)\|) + \|\mathbf{I}(D\,;\mathcal{H})\|)$.

4.1 Samples have Outliers

In many areas, one needs to obtain numbers or infinite sequences b that are "typical" with respect to a computable measure P. Being typical with respect to P means having a small ***deficiency of randomness***, $\mathbf{d}(b|P) \stackrel{\text{def}}{=} \|P(b)\| - \mathbf{K}(b)$. Atypical elements b with large $\mathbf{d}(b|P) \gg 0$ have extra regularities that allow them to be compressed to length $\mathbf{K}(b)$, which is much less than $\|P(b)\|$. With a little luck, it is not difficult to produce elements b with $\mathbf{d}(b|P) \approx 0$, or small sets of such elements. However, we show that only exotic samples $D \subset \mathbb{N}$ of large size $\|D\| = 2^n$ have all elements $b \in D$ typical of P, with $\mathbf{d}(b|P) = o(n)$. Thus *non-exotic* P-samples will always contain P-atypical elements.

Corollary 2. *Given computable* $P \in \mathscr{P}(\mathbb{N})$*, for any finite set* $D \subseteq \mathbb{N}$,
$\log \sum_{a\in D} 2^{\mathbf{d}(a|P)} \lesssim \max_{a\in D}(\mathbf{d}(a|P)) + \mathbf{I}(D\,;\mathcal{H})$.

Theorem 4 (Epstein and Hescott and Homer (2011))**.**
Given computable $P \in \mathscr{P}(\Sigma^\infty)$*, for any compact set* $Z \subseteq \Sigma^\infty$*, if* $s < \log \sum_{\alpha\in Z} 2^{\mathbf{d}(\alpha|P)}$*, then* $s \leq \sup_{\alpha\in Z}(\mathbf{d}(\alpha|P)) + \mathbf{I}(Z_{<\infty} : \mathcal{H}) + O(\mathbf{K}(s) + \|\mathbf{I}(Z_{<\infty} : \mathcal{H})\|)$.

Proof. For $x \in \Sigma^*$, $\mathbf{d}^b(x|P) \stackrel{\text{def}}{=} \max_{y\sqsubseteq x}(\log(\mathbf{m}^b(y)/P(y)))$. Let b be the shortest total string with $\sum_{x\in Z_{\leq N}} 2^{\mathbf{d}^b(x|P)} > 2^s$, where $N \stackrel{\text{def}}{=} \mathbf{bb}(b)$. We let $W(x) \stackrel{\text{def}}{=} 2^s P(x)$, $\eta(x) \stackrel{\text{def}}{=} 2^{\mathbf{d}^b(x|P)-s}[x \in \Sigma^N]$. $D \stackrel{\text{def}}{=} Z_{\leq N}$. So $\mathbf{K}(\langle W, \eta\rangle|\langle b, s\rangle) = O(1)$, $\sum_{x\in D} \eta(x) \geq 1$, and $\sum_{x\in\Sigma^*} W(x)\eta(x) = \sum_{x\in\Sigma^N} P(x)2^{\mathbf{d}^b(x|P)} = \int_\alpha 2^{\mathbf{d}^b(\alpha_{\leq N}|P)} dP(\alpha) \leq \int_\alpha 2^{\mathbf{d}(\alpha_{\leq N}|P)} dP(\alpha) \leq \int_\alpha 2^{\mathbf{d}(\alpha|P)} dP(\alpha) \leq 1$. Lemma 2, relativized to $\langle b, s\rangle$, gives $x \in D$ with $\mathbf{K}(x|\langle b, s\rangle) \prec -\log W(x) + \Lambda(D|\langle b, s\rangle) = \|P(x)\| - s + \Lambda(D|\langle b, s\rangle)$. Let $h_x \stackrel{\text{def}}{=} \mathbf{I}(D\,;\mathcal{H}|x)$.

Due to lemma 1, $\mathbf{K}(x|\langle b,s\rangle) \leq -s + \|P(x)\| + h_{\langle b,s\rangle} + O(\log h_{\langle b,s\rangle})$. So $s \leq \log(\mathbf{m}(x)/P(x)) + \mathbf{K}(b) + h_b + O(\mathbf{K}(s) + \log h_b)$. By proposition 1, $\mathbf{K}(b) + h_b \lesssim h_\emptyset + \mathbf{K}(b|\langle D, \|b\|\rangle)$. Therefore $s \leq \log(\mathbf{m}(x)/P(x)) + h_\emptyset + \mathbf{K}(b|\langle D, \|b\|\rangle) + O(\mathbf{K}(s) + \log(h_\emptyset + \mathbf{K}(b|\langle D, \|b\|\rangle)))$.

Since $D \subseteq \Sigma^{\mathbf{bb}(b)}$, $\mathbf{K}(b|\langle D, \|b\|\rangle) = O(1)$. Hence $s \leq \log(\mathbf{m}(x)/P(x)) + h_\emptyset + O(\mathbf{K}(s) + \log h_\emptyset)$. Due to proposition 1, $\|b\| \lesssim h_\emptyset + \mathbf{K}(b|\langle D, \|b\|\rangle)$. So $\|b\| \lesssim h_\emptyset$. $\mathbf{K}(D|Z_{<\infty}) \prec \mathbf{K}(\langle \|b\|, s\rangle)$, as D is computable from $Z_{<\infty}$, $\|b\|$, and s. By the definition of the extension of mutual information to infinite sequences, $\mathbf{K}(D) - \mathbf{K}(D|\mathcal{H}) - \mathbf{K}(D|Z_{<\infty}) \leq \mathbf{I}(Z_{<\infty} : \mathcal{H})$. $h_\emptyset \prec \mathbf{I}(Z_{<\infty} : \mathcal{H}) + \mathbf{K}(D|Z_{<\infty}) \prec \mathbf{I}(Z_{<\infty} : \mathcal{H}) + \mathbf{K}(\langle \|b\|, s\rangle) \lesssim \mathbf{I}(Z_{<\infty} : \mathcal{H}) + \mathbf{K}(s) + \|h_\emptyset\|$. So $h_\emptyset \lesssim \mathbf{I}(Z_{<\infty} : \mathcal{H}) + \mathbf{K}(s)$. So $s \leq \log(\mathbf{m}(x)/P(x)) + h_\emptyset + O(\mathbf{K}(s) + \|h_\emptyset\|) \leq \sup_{\alpha \in Z}(\mathbf{d}(\alpha|P)) + \mathbf{I}(Z_{<\infty} : \mathcal{H}) + O(\mathbf{K}(s) + \|\mathbf{I}(Z_{<\infty} : \mathcal{H})\|)$. □

Corollary 3. *Given computable $P \in \mathscr{P}(\Sigma^\infty)$, for any compact set $Z \subseteq \Sigma^\infty$,* $\log \sum_{\alpha \in Z} 2^{\mathbf{d}(\alpha|P)} \lesssim \sup_{\alpha \in Z}(\mathbf{d}(\alpha|P)) + \mathbf{I}(Z_{<\infty} : \mathcal{H})$.

4.2 Complexity of Classification

In machine learning, classification is the problem of inferring an unknown predicate Υ over $\mathbb{N}$ using training information. The training data is γ, a predicate with a finite domain and with $\gamma(i) = \Upsilon(i)$ for all $i \in \mathrm{Dom}(\gamma)$. Learning algorithms use γ to compute a complete predicate (hypothesis) h over $\mathbb{N}$ that approximates the hidden concept Υ. With certain probabilistic assumptions, learning algorithms that produce hypotheses $h \supseteq \gamma$ of low Kolmogorov complexity are likely to approximate Υ well [Blumer et al. (1989)].

For each $i \in \mathbb{N}$, there exists a predicate ξ_i with a domain of two 2^i-bit integers and with no complete extension of complexity $<i$. Corollary 4 shows such predicates ξ_i are *exotic*. For *non-exotic* predicates γ, the smallest encoding length of an extension $h \supseteq \gamma$ is close to the size of the domain of γ.

Theorem 5. *For any finite prefix-free set G of strings,*
$\mathbf{Km}(G) \leq \mathbf{KM}(G) + \mathbf{I}(G\,;\mathcal{H}) + O(\mathbf{K}(\mathbf{KM}(G)) + \|\mathbf{I}(G\,;\mathcal{H})\|)$.

Proof. Let $i \stackrel{\text{def}}{=} \mathbf{KM}(G)$ and N' be the smallest number with $> 2^{-i}$ fraction $D' \subseteq \Sigma^{N'}$ of inputs x such that $\nu_{\mathbf{M}}(x) \in G\Sigma^*$. Let b be the shortest total string with $N \stackrel{\text{def}}{=} \mathbf{bb}(b) \geq N'$. Let $D'' = \Sigma^N \cap \nu_{\mathbf{M}}^{-1}(G\Sigma^*)$. $\mathbf{K}(D''\,|\langle G, b\rangle) = O(1)$.

Let $W((x,J)) \stackrel{\text{def}}{=} 2^{-i}$ and $\eta((x,J)) = 2^{i-N}[x \in \Sigma^N]$, where $\mathbf{K}(\langle W, \eta\rangle|\langle b, i\rangle) = O(1)$. Let $D \stackrel{\text{def}}{=} D'' \times \{\langle G\rangle\}$. Lemma 2, relativized to $\langle b, i\rangle$, gives $a = (x, \langle G\rangle) \in D$ with $\mathbf{K}(x|\langle b, i\rangle) \prec -\log W((x, \langle G\rangle)) + \Lambda(D|\langle b, i\rangle) = i + \Lambda(D|\langle b, i\rangle)$. Due to lemma 1, $\mathbf{K}(x|\langle b, i\rangle) \leq i + \mathbf{I}(D\,;\mathcal{H}|\langle b, i\rangle) + O(\log \mathbf{I}(D\,;\mathcal{H}|\langle b, i\rangle))$. Let $h_x \stackrel{\text{def}}{=} \mathbf{I}(G; \mathcal{H}|x)$.

$\mathbf{K}(G|\langle b, D, i\rangle) \asymp \mathbf{K}(D|\langle b, G, i\rangle) = O(1)$. So $\mathbf{K}(x|\langle b, i\rangle) \leq i + h_{\langle b,i\rangle} + O(\log h_{\langle b,i\rangle})$. So $\mathbf{K}(x) \leq i + \mathbf{K}(b) + h_b + O(\mathbf{K}(i) + \log h_b)$. By proposition 1, $\mathbf{K}(b) + h_b \lesssim h_\emptyset + \mathbf{K}(b|\langle G, \|b\|\rangle)$. Hence $\mathbf{K}(x) \leq i + h_\emptyset + \mathbf{K}(b|\langle G, \|b\|\rangle) + O(\mathbf{K}(i) + \log(h_\emptyset + \mathbf{K}(b|\langle G, \|b\|\rangle)))$. $\mathbf{K}(b|\langle G, \|b\|\rangle) \prec \mathbf{K}(i)$. This means $\mathbf{K}(x) \leq i + h_\emptyset + O(\mathbf{K}(i) + \log h_\emptyset)$. $\mathbf{Km}(G) \prec \mathbf{K}(x)$ because $\nu_{\mathbf{M}}(x) \in G\Sigma^*$. So $\mathbf{Km}(G) \leq i + h_\emptyset + O(\mathbf{K}(i) + \log h_\emptyset)$. □

Corollary 4. *For predicate γ over* $\mathrm{Dom}(\gamma) \subset \mathbb{N}$, $\min_{h \supseteq \gamma} \mathbf{K}(h) \lesssim \|\mathrm{Dom}(\gamma)\| + \mathbf{I}(\gamma; \mathcal{H})$.

Proof. Let G be a set of strings x with $\|x\| = \max\{i \in \mathrm{Dom}(\gamma)\}$ and $x \supseteq \gamma$. Thus $\mu(G) = 2^{-\|\gamma\|}$. Theorem 5, applied to G, gives $h \in G\Sigma^*$ where $\mathbf{K}(h) \lesssim \mathbf{KM}(G) + \mathbf{I}(G; \mathcal{H}) \prec \|\mu(G)\| + \mathbf{I}(\gamma; \mathcal{H}) = \|\mathrm{Dom}(\gamma)\| + \mathbf{I}(\gamma; \mathcal{H})$. □

The predicate h constructed in the above proof has a finite domain. Also, corollary 4 is tight.

Lemma 3. *For all $d, i \in \mathbb{N} \setminus \{0,1,2\}$, there is a predicate γ where (1) $\|\mathrm{Dom}(\gamma)\| = d$, (2) $\mathbf{I}(\gamma;\mathcal{H}) = i \pm O(\log(di))$, and (3) $\min_{h \supseteq \gamma} \mathbf{K}(h) \sim d + i$.*

Proof. Let r be a $d-3$ bit random string such that $\mathbf{I}(r;\mathcal{H}) \leq \log(di)$. Let p be the rightmost total i-bit string with respect to r. So $\mathbf{I}(p;\mathcal{H}|r) \sim i$. There exists a predicate ξ such that $\min_{h \supseteq \xi} \mathbf{K}(h|r) = i \pm O(\log(di))$ and $\mathbf{K}(\xi|\langle p, r\rangle) + \mathbf{K}(p|\langle r, \xi\rangle) = O(\log(di))$. Indeed, let $A \subset \{0,1,\perp\}^k$, $k = 3^{2^i}+d$, be the set of initial segments of all predicates γ of complexity $\mathbf{K}(\gamma|r) \leq i - \mathbf{K}(\langle d, i\rangle|r) - c$, where c is a constant solely dependent on the universal algorithm U_r. A can be computed from $\langle d, i, p\rangle$. Since $\|A\| < 2^i$, there exists distinct $s, t \in [d, 3^{2^i}+d]$ such that $x_s = x_t$ for all $x \in A$. Therefore the predicate $\xi \stackrel{\text{def}}{=} \{(\mathbf{bb}(p|r), 0), (s, 0), (t, 1)\}$ satisfies the properties stated above. $\gamma \stackrel{\text{def}}{=} r \cup \xi$. The property (1) follows because $\|\mathrm{Dom}(\gamma)\| = d$.

Property (2) follows because $\mathbf{I}(\gamma\,;\mathcal{H}) \asymp \mathbf{I}(\langle r, \xi\rangle;\mathcal{H}) = \mathbf{K}(\langle r, \xi\rangle) - \mathbf{K}(\langle r, \xi\rangle|\mathcal{H}) = \mathbf{K}(r) + \mathbf{K}(\xi|r) - \mathbf{K}(\xi|\langle r, \mathcal{H}\rangle) - \mathbf{K}(r|\mathcal{H}) \pm O(\log d) = \mathbf{I}(r;\mathcal{H}) + \mathbf{I}(\xi;\mathcal{H}|r) \pm O(\log d) = \mathbf{I}(r;\mathcal{H}) + \mathbf{I}(p;\mathcal{H}|r) \pm O(\log(di)) = i \pm O(\log(di))$.

Property (3) holds as well. Indeed, let $h \stackrel{\text{def}}{=} \arg\min_{h' \supseteq \gamma} \mathbf{K}(h')$. By the chain rule, $\mathbf{K}(r) + \mathbf{K}(h|\langle r, \mathbf{K}(r)\rangle) \asymp \mathbf{K}(h) + \mathbf{K}(r|\langle h, \mathbf{K}(h)\rangle)$. So $\mathbf{K}(r) + \mathbf{K}(h|r) - \mathbf{K}(r|h) \leq \mathbf{K}(h) + O(\mathbf{K}(\mathbf{K}(h), \mathbf{K}(r))) \lesssim \mathbf{K}(h) + O(\log d)$. $\mathbf{K}(r|h) \prec \mathbf{K}(d)$ since $r \subseteq h$, and $\mathbf{K}(h|r) \geq i - O(\log(di))$ since $\xi \subseteq h$. Thus $\mathbf{K}(r) + i \lesssim \mathbf{K}(h) + O(\log(di))$. Therefore $d + i \lesssim \mathbf{K}(h)$. In addition, $\mathbf{K}(\gamma) \prec \mathbf{K}(\langle \xi, i, r\rangle) \asymp \mathbf{K}(\langle p, r\rangle) \lesssim d + i$ and $\gamma \subseteq \gamma$. So $\min_{h \supseteq \gamma} \mathbf{K}(h) \sim d + i$. □

4.3 Uncomputability of K

Corollary 5 shows that an encoding of any 2^n unique pairs $\langle b, \mathbf{K}(b)\rangle$ has more than $\sim n$ bits of mutual information with the halting sequence $\mathcal{H}$. So all such large sets are *exotic*.

Theorem 6. *For any finite set T of natural numbers and $L:T\to\mathbb{N}$,*
$s \stackrel{def}{=} \lfloor \log \|T\| \rfloor \leq 2\max_{a\in T}(|L(a)-\mathbf{K}(a)|) + \mathbf{I}(L:\mathcal{H}) + O(\mathbf{K}(s)+\|\mathbf{I}(L:\mathcal{H})\|)$.

Proof. Let $j \stackrel{\text{def}}{=} \max_{a\in T}|L(a)+\lceil \log \mathbf{m}(a)\rceil|$. Note $\mathbf{K}(a) \asymp -\log \mathbf{m}(a)$. Let b be the shortest total string with $\max_{a\in T}|L(a)+\lceil \log \mathbf{m}^b(a)\rceil| \leq j$. So for all $a\in T$, $\|\mathbf{m}^b(a)\| - \mathbf{K}(a) \prec 2j$. Let $D \stackrel{\text{def}}{=} T\times\{\langle L\rangle\}$, $\eta((a,j)) = 2^{-s}$, and $W((a,j)) \stackrel{\text{def}}{=} 2^s\mathbf{m}^b(a)$.

$\mathbf{K}(\langle W,\eta\rangle|\langle b,s\rangle) \stackrel{\text{def}}{=} O(1)$. Lemma 2, relativized to $\langle b,s\rangle$ gives $(a,\langle L\rangle)\in D$ where $\mathbf{K}(a|\langle b,s\rangle) \prec -\log \mathbf{m}^b(a) - s + \Lambda(D|\langle b,s\rangle)$. So $s \prec -\log \mathbf{m}^b(a) - \mathbf{K}(a|\langle b,s\rangle) + \Lambda(D|\langle b,s\rangle) \prec -\log \mathbf{m}^b(a) - \mathbf{K}(a) + \mathbf{K}(\langle b,s\rangle) + \Lambda(D|\langle b,s\rangle) \asymp \log(\mathbf{m}(a)/\mathbf{m}^b(a)) + \mathbf{K}(\langle b,s\rangle) + \Lambda(D|\langle b,s\rangle) \leq 2j + \mathbf{K}(\langle b,s\rangle) + \Lambda(D|\langle b,s\rangle)$. Due to lemma 1, $s \leq 2j + \mathbf{K}(\langle b,s\rangle) + \mathbf{I}(D;\mathcal{H}|\langle b,s\rangle) + O(\log \mathbf{I}(D;\mathcal{H}|\langle b,s\rangle))$. Let $h_x \stackrel{\text{def}}{=} \mathbf{I}(L;\mathcal{H}|x)$.

Since $\mathbf{K}(D|L) \asymp \mathbf{K}(L|D) = O(1)$, $s \leq 2j + \mathbf{K}(\langle b,s\rangle) + h_{\langle b,s\rangle} + O(\log h_{\langle b,s\rangle})$. So $s \leq 2j + \mathbf{K}(b) + h_b + O(\mathbf{K}(s) + \log h_b)$. Due to proposition 1, $\mathbf{K}(b) + h_b \lesssim h_\emptyset + \mathbf{K}(b|\langle L,\|b\|\rangle)$. $\mathbf{K}(b|\langle L,\|b\|\rangle) \prec \mathbf{K}(j)$. So $\mathbf{K}(b) + h_b \lesssim h_\emptyset + \mathbf{K}(j)$. $s \leq 2j + h_\emptyset + O(\mathbf{K}(s) + \mathbf{K}(j) + \log h_\emptyset)$. Assuming $s > 2j$, we have $\mathbf{K}(s-2j) \leq O(\|s - 2j\|) \leq O(\|\mathbf{K}(s)+\mathbf{K}(j)+h_\emptyset\|)$. So $\mathbf{K}(j) \prec \mathbf{K}(s)+\mathbf{K}(s-2j) \leq O(\mathbf{K}(s)+\|\mathbf{K}(j)+h_\emptyset\|)$. Therefore $\mathbf{K}(j) \leq O(\mathbf{K}(s)+\|h_\emptyset\|)$. So $s \leq 2j + h_\emptyset + O(\mathbf{K}(s)+\|h_\emptyset\|)$. □

Corollary 5. *Any set $X\subset\Sigma^*$ of 2^n unique pairs $\langle b,\mathbf{K}(b)\rangle$ has $n \lesssim \mathbf{I}(X:\mathcal{H})$.*

4.4 Exotic Functions

The following theorem shows that any non-negative function f high correlated to $\mathbf{m}$ is exotic.

Theorem 7. *For any function $f:\mathbb{N}\to\mathbb{N}$ of finite support T, where $f(a)\mathbf{m}(a) > 1$ for all $a\in T$, $\log\sum_{a\in T} f(a)\mathbf{m}(a) \lesssim \log\max_{a\in T}(f(a)\mathbf{m}(a)) + \mathbf{I}(f:\mathcal{H})$.*

Proof. Let $s \stackrel{\text{def}}{=} \lceil \log\sum_{a\in T} f(a)\mathbf{m}(a)\rceil - 1$ and $j \stackrel{\text{def}}{=} \max_{a\in T}\lceil \log f(a)\mathbf{m}(a)\rceil$. Let b be the shortest total string with $\log\sum_{a\in T} f(a)\mathbf{m}^b(a) > s$ and $f(a)\mathbf{m}^b(a) > 1$, for all $a\in T$.

Let $z \stackrel{\text{def}}{=} \langle j, s\rangle$. Let $S \stackrel{\text{def}}{=} \text{supp}(\mathbf{m}^b)$ and $\mathcal{G}$ be the set of all functions $g : S \to \mathbb{N} \setminus \{0\}$. Let $\kappa(g) \in \mathscr{P}(\mathcal{G})$ be a probabilistic measure over $\mathcal{G}$ where $\kappa(g) \stackrel{\text{def}}{=} \prod_{a\in S} 2^{-g(a)}$. Let $\mathcal{G}_1 \subseteq \mathcal{G}$ be the set of functions $g \in \mathcal{G}$ where there exists $a \in T$ with $g(a)=s-\lceil \log f(a)\mathbf{m}^b(a)\rceil-1$.

Using $(1-m)e^m \leq 1$ for $m \in [0,1]$ we get the inequality, $\kappa(\mathcal{G}\backslash\mathcal{G}_1) \leq \prod_{a\in T}(1-2f(a)\mathbf{m}^b(a)2^{-s}) \leq e^{-2*2^{-s}\sum_{a\in T} f(a)\mathbf{m}^b(a)} \leq e^{-2}$. Thus $\kappa(\mathcal{G}_1) > 0.75$. We use measures $P_g(a) \in \mathscr{M}(S)^+$ indexed by $g \in \mathcal{G}$, defined as $P_g(a) \stackrel{\text{def}}{=} [g(a)\leq s]\mathbf{m}^b(a)2^{g(a)}/2s$.

Therefore $\mathbf{E}_\kappa[P_g(S)]=\mathbf{E}_\kappa[\sum_a P_g(a)] = \mathbf{E}_\kappa[\sum_a \mathbf{m}^b(a)\left[g(a)\leq s\right]2^{g(a)}/2s\big] = \sum_a \mathbf{m}^b(a)$ $\mathbf{E}_\kappa\left[[g(a)\leq s]2^{g(a)}\right]/2s = \sum_a \mathbf{m}^b(a)\sum_{n=1}^s 2^n\kappa(\{g : g(a)=n, g\in\mathcal{G}\})/2s = \sum_a \mathbf{m}^b(a)$ $\sum_{n=1}^s 2^n 2^{-n}/2s = .5\sum_a \mathbf{m}^b(a) \leq 0.5$. Let $\mathcal{G}_2 \stackrel{\text{def}}{=} \{g : P_g(S) \leq 1, g \in \mathcal{G}\}$. So $\kappa(\mathcal{G}_2) \geq 0.5$ and $\kappa(\mathcal{G}_1 \cap \mathcal{G}_2) > 0.25$. Let $D' \subset \mathcal{G}_1 \cap \mathcal{G}_2$ be a minimal set such that $\kappa(D') \geq 0.25$. $\mathbf{K}(D'|\langle b, f, z\rangle) = O(1)$.

Let $D \stackrel{\text{def}}{=} D' \times \{\langle f\rangle\}$, $\eta((g,j)) \stackrel{\text{def}}{=} 4\kappa(g)$, and $W((g,j)) \stackrel{\text{def}}{=} 0.25$. $\mathbf{K}(\langle W, \eta\rangle|\langle b, z\rangle)=O(1)$. Lemma 2, relativized to $\langle b, z\rangle$, gives $(g, \langle f\rangle) \in D$ with $\mathbf{K}(g|\langle b, z\rangle) \prec -\log W(g) + \Lambda(D|\langle b, z\rangle) \succ \Lambda(D|\langle b, z\rangle)$. By lemma 1, $\mathbf{K}(g|\langle b, z\rangle) \lesssim \mathbf{I}(D; \mathcal{H}|\langle b, z\rangle) \succ \mathbf{I}(f; \mathcal{H}|\langle b, z\rangle)$.

Since $g \in \mathcal{G}_2$, $P_g(S) \leq 1$. So for all $x \in S$, $\mathbf{K}(x|\langle g, b, z\rangle) \prec -\log P_g(x) + \mathbf{K}(P_g|\langle g, b, z\rangle)$ $\succ -\log P_g(x)$. Now $\mathbf{K}(x|\langle b, z\rangle) \prec \mathbf{K}(x|\langle g, b, z\rangle) + \mathbf{K}(g|\langle b, z\rangle) \prec -\log P_g(x) + \mathbf{K}(g|\langle b, z\rangle)$. Let $h_x \stackrel{\text{def}}{=} \mathbf{I}(f; \mathcal{H}|x)$. Since $\mathbf{K}(g|\langle b, z\rangle) \lesssim h_{\langle b,z\rangle}$, it must be that $\mathbf{K}(x|\langle b, z\rangle) \leq -\log P_g(x)$ $+h_{\langle b,z\rangle} + O(\log h_{\langle b,z\rangle})$. So $\mathbf{K}(x|z) \leq -\log P_g(x) + \mathbf{K}(b|z) + h_{\langle b,z\rangle} + O(\log h_{\langle b,z\rangle})$. By proposition 1 relativized to z, for all $x \in S$, $\mathbf{K}(x|z) \leq -\log P_g(x) + h_z + \mathbf{K}(b|\langle f, z, \|b\|\rangle) +$ $O(\log(h_z + \mathbf{K}(b|\langle f, z, \|b\|\rangle)))$.

$\mathbf{K}(b|\langle f, z, \|b\|\rangle) = O(1)$. So $\mathbf{K}(x|z) \leq -\log P_g(x) + h_z + O(\log h_z)$. Since $g \in \mathcal{G}_1$, there exists $a \in T$, where $g(a) = s - \lceil \log f(a)\mathbf{m}^b(a)\rceil - 1$. So $-\log P_g(a) \succ -\log \mathbf{m}^b(a) - g(a) + \log s \succ \log f(a) - s + \log s$. Since $a \in S$, $\mathbf{K}(a|z) \leq \log f(a) - s + h_z + O(\log s + \log h_z)$. Since $\mathbf{K}(z) = O(\log s)$, $\mathbf{K}(a) \succ -\log \mathbf{m}(a) \leq \log f(a) - s + h_\emptyset + O(\log s + \log h_\emptyset)$. So $s \lesssim \log f(a)\mathbf{m}(a) + h_\emptyset \lesssim j + h_\emptyset$. $\square$

Chapter 5

Learning With Kernels

5.1 Introduction

We consider the problem of approximating a computationally expensive distance of a real-time observation to a set of training data. Given is a set of n samples $Q = \{q_1, q_2, \ldots, q_n\}$, from a sample space $\mathcal{Q}$, with $Q \subset \mathcal{Q}$, representing training data and a new sample q, representing a real-time observation. Also given is the target distance metric d which takes $\Theta(e)$ time to compute. The goal is to approximate the distance $d(q, q_i)$ of the real-time observation q to each $q_i \in Q$ in $o(en)$ time.

Our posed problem of sparse distance approximation represents a variant in the general literature of distance metric learning [Yang and Jin (2006)]. The general problem of distance metric learning is common in real-world applications such as computer vision and content retrieval.

We present a general solution to the sparse distance approximation problem, relying only on the assumption that the target distance metric d is Hilbertian. We review the Semi-least Squares problem, introduced by [Radhakrishna and Mitra (1971)], in Section 5.2. We review kernel literature in Section 5.3. The contribution of this thesis is to show the connection between these two ideas. By extending the Semi-least Squares problem to the Kernel Semi-least Squares problem, we provide a computationally advantageous method to solving the important problem of distance approximation (Sections 5.5 and 5.6). In Section 5.7, we provide a method for choosing the best training subset for distance approximation. We also provide an alternate derivation of

the solution (Section 5.8). Related works are discussed in Section 5.9. Experimental results of our method and a comparison to the Nyström method [Williams and Seeger (2001)] are shown in Section 5.10.

5.2 The Semi-least Squares Problem

In this section, we describe the Semi-least Squares problem introduced by [Radhakrishna and Mitra (1971)]. In this problem, the traditional L2-norm $\|\cdot\|$, used by the well-known Least Squares problem, is replaced by a seminorm. A seminorm, $\rho(\cdot)$, differs from a norm in that it is permitted that $\rho(u) = 0$ for some non-zero vectors u.

A square matrix J is positive semidefinite if it allows a decomposition[1] $J = HH^*$, for some matrix H. We define seminorm $\|z\|_J = (z^*Jz)^{1/2}$, where $J = HH^*$ is a positive semidefinite matrix. It is not a proper norm because for all vectors v in the null space of H, $\|v\|_J = 0$. A matrix G is said to be a *Semi-least Squares* inverse of a matrix A if the minimum of

$$\|Az - y\|_J \tag{5.1}$$

is attained at $z = Gy$, for any y. Such a G exists, being of the form

$$G \;=\; (A^*JA)^{-1}A^*J + P. \tag{5.2}$$

The matrix P is any projection onto the null space of H^*A. The precise form of P is $[I - (A^*JA)^+(A^*JA)]\,U$, for any matrix U. The $(\cdot)^+$ operation represents the Moore-Penrose pseudoinverse.

Computation efficiency was not discussed in Rao and Mitra's 1971 paper. However computational benefits emerge when the Semi-least Squares problem is extended with kernels, as shown in Section 5.6.

[1] The * symbol represents the transpose operation.

5.3 Kernels

We assume the distance function d is Hilbertian. A distance metric d is Hilbertian if it can be embedded into a vector space, with finite or infinite number of dimensions. We can use the inner product function (or kernel function) associated with this space to perform useful functions, such as projections. We define such a kernel function k by

$$k(q, q') = g(q) - \frac{1}{2}d^2(q, q') + g(q'), \tag{5.3}$$

for any function $g : \mathcal{Q} \to \mathbb{R}^+$. One common form is $g(q) = \frac{1}{2}d^2(q, q')$ for some $q' \in \mathcal{Q}$. Since d is Hilbertian, k is semi-positive definite. This implies the kernel function k represents the inner product over Q, mapped to a Hilbert space $\mathcal{F} = \mathbb{R}^l$, with $l \in \mathbb{N} \cup \{\infty\}$. This mapping from $\mathcal{Q}$ to $\mathcal{F}$ is represented by the function $\phi(q) : \mathcal{Q} \to \mathcal{F}$, with

$$k(q, q') = \phi^*(q)\phi(q'). \tag{5.4}$$

The kernel substitution method, known also as the "kernel trick", allows the mapping ϕ to be implicitly defined by kernel k [Schölkopf and Smola (2001)].

We introduce some standard definitions associated with kernels. The Gram matrix K and the design matrix Φ are defined with respect to the set $Q = \{q_1, q_2, \ldots, q_n\}$, and a mapping ϕ. The Gram matrix K contains all the inner products between mappings of elements of Q. It is an $n \times n$ matrix, whose (i, j)th element is $k(q_i, q_j)$. The Gram matrix is an explicit construct. The design matrix Φ is a listing of the mapping of the elements of Q using ϕ. The design matrix Φ is an $l \times n$ matrix whose ith column is $\phi(q_i)$.

The Gram matrix and the design matrix are related by $\Phi^*\Phi = K$. The empirical

map is denoted by $\mathbf{k}_Q : \mathcal{Q} \to \mathbb{R}^n$, where the ith value of vector $\mathbf{k}_Q(q)$ is $k(q_i, q)$, with $q_i \in Q$. It follows that $\Phi^* \phi(q) = \mathbf{k}_Q(q)$. The Gram matrix K and empirical map $\mathbf{k}_Q(\cdot)$ can be computed, whereas the design matrix Φ and the mapping $\phi(\cdot)$ are not generally computable.

5.4 Distance Decomposition

In this section we show how to compute the distance $d(q, q_i)$, with $q_i \in Q$, using tangent and orthogonal components with respect to the linear span of Q. This decomposition of the distance d into orthogonal components is a convenient form to be used in Section 5.5 for sparse distance approximation. Let k be a kernel function whose induced distance is d, with $d^2(q, q') = k(q, q) + k(q', q') - 2k(q, q')$. The kernel k defines a kernel feature space $\mathcal{F}$ whose distances are congruent with d, and an implicit mapping $\phi : \mathcal{Q} \to \mathcal{F}$. The linear span of the set Q in $\mathcal{F}$ is defined by $\{\sum \alpha_i \phi(q_i) \mid \alpha \in \mathbb{R}^n\}$.

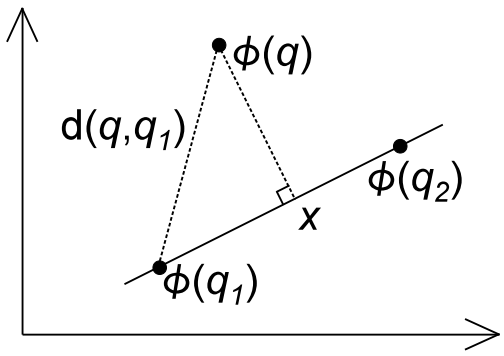

Figure 5·1: The point x represents the projection of the observation $\phi(q)$ onto the linear span of the set Q in kernel space $\mathcal{F}$. The vector from q to q_1 can be seen as the decomposition into two orthogonal components, the vector from $\phi(q)$ to x and the vector of x to $\phi(q_1)$. The linear span intersects the origin of $\mathcal{F}$.

Let x represent the results of a projection from the mapped observation $\phi(q)$ to the span of Q in $\mathcal{F}$, as seen in Figure 5·1. Since x is the result of an orthogonal

projection, the distance function $d(q, q_i)$ for each $q_i \in Q$ can be defined by orthogonal components, with

$$d(q, q')^2 \quad = \quad \|\phi(q) - x\|^2 + \|x - \phi(q')\|^2. \tag{5.5}$$

In the rest of this section, we show how to compute the two terms of Equation 5.5.

Kernel Projection

The point x can be defined according to the L2 norm, with

$$x = \arg \min_{x' \in \text{Span}(Q) \text{ in } \mathcal{F}} \|x' - \phi(q)\|. \tag{5.6}$$

In general, the point x cannot be explicitly computed, since $\mathcal{F}$ has potentially infinite dimension. However point x can be represented as the linear combination of the observations of training set Q, mapped into Φ,

$$x \quad = \quad \sum_{i=1}^{n} \beta_i \phi(q_i) = \Phi\, \beta, \tag{5.7}$$

where β_i is the ith element of the vector $\beta \in \mathbb{R}^n$ that represents the coordinates of x using $\phi(q_i)$ as a basis. The projection used to produce point x can be formally defined using the Least Squares methodology and the coordinates β. A mapping $h(q) = \beta$ is defined to be the *Kernel Least Squares* inverse of a training set Q if

$$h(q) \quad = \quad \arg\min_{\beta} \sum_{q \in Q} \|\Phi\beta - \phi(q)\|.$$

Such a mapping exists and can be derived using Least Squares methodology, with

$$h(q) = \beta = \Phi^{+}\phi(q) = \left(\Phi^{*}\Phi\right)^{-1} \Phi^{*}\phi(q) = K^{-1}\mathbf{k}_Q(q), \tag{5.8}$$

where the term Φ^+ represents the pseudoinverse of the design matrix (A similar form of Equation 5.8 was described by [Schölkopf et al. (1999)]. This mapping is known as the *kernel projection*. The Gram matrix K might be singular, but the limit $\lim_{\delta \to 0}(K + \delta I)^{-1}\mathbf{k}_Q(q)$ is guaranteed to exist by the definition of pseudoinverses.

Computation of Distance Decomposition

The coordinates β representing point x can be used to compute both distances of Equation 5.5. The term $\|\phi(q) - x\|^2$ can be computed using the fact that the span of Q in $\mathcal{F}$ contains the origin, and x is a orthogonal projection, with

$$\begin{aligned} \|\phi(q) - x\|^2 &= \|\phi(q)\|^2 - \|x\|^2 = \phi(q)^*\phi(q) - \beta^*\Phi^*\Phi\beta \\ &= k(q,q) - \beta^* K \beta. \end{aligned} \tag{5.9}$$

The term $\|x - \phi(q_i)\|^2$ can be computed by a direct substitution, with

$$\begin{aligned} \|x - \phi(q_i)\|^2 &= \|\Phi\,\beta - \phi(q_i)\|^2 \\ &= \beta^* K \beta - 2\beta^* K_i + K_{i,i}. \end{aligned} \tag{5.10}$$

The term K_i represents the ith column of the Gram matrix K of Q. Equations 5.9 and 5.10 can be substituted back into the distance decomposition of Equation 5.5, resulting in Equation 5.12. Given a kernel function k, a set Q with associated Gram matrix K, the distance $d(q, q_i)$ can be computed by the following steps:

1. The coordinates β are computed with the kernel projection,

$$\beta = K^{-1}\mathbf{k}_Q(q). \tag{5.11}$$

2. The distance $d(q, q_i)$ is computed using the coordinates, β,

$$d(q, q_i) = \left(k(q,q) - 2\beta^* K_i + K_{i,i}\right)^{1/2}. \tag{5.12}$$

5.5 Sparse Distance Approximation

Assuming a kernel of the form of Equation 6.2 is used, the time complexity of computing the kernel, $\Omega(e)$, is not less than the time complexity of computing the distance d. To compute the kernel projection of Equation 5.11, $k(q, q_i)$ needs to be computed for each $q_i \in Q$. The time complexity of this operation is on the order of $\Omega(en)$, where n is the size of the training set and assuming $n < e$. Therefore computing the decomposition of the distance into orthogonal components in Section 5.4 does not provide any time complexity benefits. However, the kernel projection of Equation 5.11 can be approximated, which, as we show below, is computationally advantageous.

We introduce the technique of *subset projection*, which approximates the kernel projection using a secondary set of observations $R = \{r_1, r_2, \ldots, r_m\}$, where $|R| = m$, $|Q| = n$, and typically $R \subset Q$ and $m \ll n$. It is of the form

$$\hat{\beta} \quad = \quad (K_{RQ}^{+} + W)\mathbf{k}_R(q). \tag{5.13}$$

The term K_{RQ} is an $m \times n$ matrix whose value at position (i, j) is equal to $k(r_i, q_j)$. This matrix can be informally thought of as a "cross" Gram matrix between R and Q. The $m \times n$ matrix W represents any projection onto the null space of K_{RQ}. The term $\mathbf{k}_R$ represents the empirical function for R.

Whereas the kernel projection minimizes an L2 norm and provides a Kernel Least Squares solution, the subset projection of Equation 5.13 minimizes a seminorm and provides a *Kernel Semi-least Squares* solution. The Kernel Semi-least Squares problem is defined in Section 5.6.

Assuming the inverse cross Gram matrix is computed offline, the computational complexity of the subset projection is $\Theta(em)$, where $n = |Q|$ and $m = |R|$, and assuming $m < n < e$. The subset projection is more efficient to compute than the kernel projection. The results of the subset projection can be used to approximate

the distance of q to each $q_i \in Q$.

Solution to Distance Approximation

Given is a kernel function k (derived from the distance d), a training set Q, and an observation q. A subset $R \subseteq Q$ is chosen, and the inverse cross matrix K^+_{RQ} and the projection W are pre-computed (the standard value of W is 0). The distance approximation, $\hat{d}$, is computed by two steps.

1. The approximate coordinates $\hat{\beta}$ are computed by the subset projection, with

$$\hat{\beta} \quad = \quad (K^+_{RQ} + W)\mathbf{k}_R(q). \tag{5.14}$$

2. The distance $\hat{d}$ is computed using the coordinates,

$$\hat{d}(q, q_i) \quad = \quad \left(k(q,q) - 2\hat{\beta}^* K_i + K_{i,i}\right)^{1/2}. \tag{5.15}$$

The approximated coordinates $\hat{\beta}$ can be reused for each $q_i \in Q$. The time complexity of the procedure is $O(em)$, assuming $m < n < e$. This implies the time complexity is $o(en)$ and thus this procedure represents a solution to the sparse distance approximation problem described in the introduction.

5.6 The Kernel Semi-least Squares Problem

We extend the Semi-least Squares problem of Section 5.2 by introducing the *Kernel Semi-least Squares* problem, for which the subset projection provides an optimal solution. Let k be a kernel, with associated mapping ϕ. Let $Q = \{q_1, q_2, \ldots, q_n\}$ be the training set and $R = \{r_1, r_2, \ldots, r_m\}$ be another set, where typically $m \ll n$ and $R \subset Q$. Sets Q and R have design matrices Φ and Φ_R, whose ith column is $\phi(q_i)$ and $\phi(r_i)$ respectively.

We define the matrix J used in the seminorm $\|\cdot\|_J$ to be the scatter matrix of R, $J = \Phi_R \Phi_R^*$. A mapping h is said to be the *Kernel Semi-least Squares* inverse of Q and R if the minimum of

$$\|\Phi\beta - \phi(q)\|_J \tag{5.16}$$

is achieved at

$$\hat{\beta} \;=\; h(q), \tag{5.17}$$

for all q. From Equation 5.2, such a mapping h exists and is of the form

$$h(q) \;=\; G\phi(q) \tag{5.18}$$

with

$$\begin{aligned} G &= (\Phi^* J \Phi)^{-1} \Phi^* J + P \\ &= (\Phi^* \Phi_R \Phi_R^* \Phi)^{-1} \Phi^* \Phi_R \Phi_R^* + P \\ &= (K_{RQ}^* K_{RQ})^{-1} K_{RQ}^* \Phi_R^* + P \\ &= K_{RQ}^+ \Phi_R^* + P. \end{aligned} \tag{5.19}$$

The term $K_{RQ} = \Phi_R^* \Phi$ is the "cross" Gram matrix. The matrix P represents any projection onto the null space of K_{RQ}. By restricting the projection P to be of the form $W\Phi_R^*$, with W being an $m \times n$ matrix that is a projection onto the null space

of K_{RQ}, we can derive a computable mapping, with

$$\begin{aligned}
\hat{\beta} &= h(q) \\
\hat{\beta} &= ((K_{RQ}^{+})\Phi_R^* + W\Phi_R^*)\phi(q) \\
&= ((K_{RQ}^{+} + W)\Phi_R^*)\phi(q) \\
&= (K_{RQ}^{+} + W)\mathbf{k}_R(q), \qquad (5.20)
\end{aligned}$$

where $\mathbf{k}_R$ representing the empirical map with respect to R. The matrix W is of the form $\left[I - K_{RQ}^{+}K_{RQ}\right]U$, for any $m \times n$ matrix U. This is the same form as the subset projection introduced in Equation 5.13. Thus the subset projection produces a solution to the Kernel Semi-least Squares problem. Assuming the matrices K_{RQ}^{+} and W are precomputed, the time complexity to compute $\hat{\beta}$ is $\Theta(nm+em) = \Theta(em)$, assuming $m < n < e$.

5.7 Subset Selection

One important open issue is how to select the subset R_o for which the corresponding approximate distance $\hat{d}_R$ is closest to the original distance d. We formalize the selection of the subset as a minimization problem, where R_o represents the optimal subset:

$$R_o = \arg\min_{R\in\mathcal{R}} \sum_{q_i\in Q} \hat{d}_R(q_i, q_i)^2. \qquad (5.21)$$

The set of candidate subsets is given by $\mathcal{R} \subseteq 2^Q$. The measure of closeness is determined by the sum of the squared approximate distances $\hat{d}_R$ of each training element $q_i \in Q$ to itself. The optimally selected subset can be rewritten as

$$R_o = \arg\max_{R\in\mathcal{R}} \sum_{i=1}^{n} \left(\hat{\beta}_i^* K_i\right), \qquad (5.22)$$

with K_i being the ith column of the Gram matrix K of Q, and $\hat{\beta}_i = K_{RQ}^{+}\mathbf{k}_R(q_i)$. This can be further simplified with

$$R_o \;=\; \arg\max_{R\in\mathcal{R},} \sum_{i=1}^{n} \left(K_{RQ}^{+}K_{RQ}K\right)_{ii}. \tag{5.23}$$

The orthogonal projection on the range of K_{RQ}^{*} is denoted by $P_{RQ} = K_{RQ}^{+}K_{RQ}$, so the final expression to compute the optimally-selected subset R_o of candidates $\mathcal{R} \subseteq 2^Q$ is

$$R_o \;=\; \arg\max_{R\in\mathcal{R}} \operatorname{Tr}\left(P_{RQ}K\right). \tag{5.24}$$

This means that the optimal subset $R_o \in \mathcal{R}$ maximizes the sum of the eigenvalues of the Gram matrix K of the training set Q, projected onto the range of $K_{R_oQ}^{*}$. A general way to recover R_o depends on the choice of the candidate subsets $\mathcal{R}$. There is future work in determining an efficient algorithm to recover R_o for different choices of the candidates. One natural set of candidates is all subsets of a certain size m, with $\mathcal{R} = \{R : R \in 2^Q, |R| = m\}$.

Minimizing the difference of distances

Another approach is to find the subset $R_o \in \mathcal{R}$ that minimizes the absolute difference of the squared distances $\hat{d}$ and d, sampled over the training set Q,

$$R_o \;=\; \arg\min_{R\in\mathcal{R}} \sum_{q_i,q_j\in Q} \left|\hat{d}_R(q_i,q_j)^2 - d(q_i,q_j)^2\right|. \tag{5.25}$$

This formulation can be reduced further with

$$R_o \;=\; \arg\min_{R\in\mathcal{R}} \sum_{i=1}^{n}\sum_{j=1}^{n} \left|(\hat{\beta}_i - \beta_i)^{*}K_j\right|, \tag{5.26}$$

with K_j being the jth column of the Gram matrix K of Q, and $\hat{\beta}_i = K_{RQ}^{+}\mathbf{k}_R(q_i)$, and $\beta_i = K^{-1}\mathbf{k}_Q(q_i)$. The expression can be further converted into a term similar to

Equation 5.24, with

$$\begin{aligned} R_o &= \arg\min_{R\in\mathcal{R}}, \sum_{i=1}^{n}\sum_{j=1}^{n} \left|(I - K_{RQ}^{+}K_{RQ}) * K\right|_{ij}, \\ &= \arg\min_{R\in\mathcal{R}} \left\|\|N_{RQ}K\|\right\|_1 . \end{aligned} \tag{5.27}$$

The orthogonal projection on the null space of K_{RQ} is represented by N_{RQ}. For this formulation, the best subset R_o minimizes the entrywise 1-norm $\|A\|_1 = \sum_{i,j} \left|a_{ij}\right|$ of the Gram matrix K of the training set Q, projected onto the nullspace of the K_{R_oQ}.

5.8 Alternate Derivation of the Subset Projection

The subset projection can also be derived as the concatenation of two sequential projections (Figure 5·2). However this derivation is more restrictive than that of Section 5.6, since it produces only one form of the subset projection with the matrix W set to zero. The first projection is from the position $\phi(q)$ to a point α in the linear span of R in feature space $\mathcal{F}$ with

$$\alpha = K_R^{-1}\mathbf{k}_R(q). \tag{5.28}$$

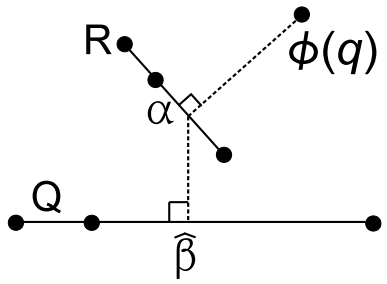

Figure 5·2: The subset projection can be interpreted as two projections. The first projection is from $\phi(q)$ to α on the span of R. The second projection is from α to $\hat{\beta}$ on the span of Q.

The second projection is from point α to the position $\hat{\beta}$ in span of Q in $\mathcal{F}$. Thus the second projection finds a position $\hat{\beta}$ that minimizes the term $\|\Phi\hat{\beta} - \Phi_R\alpha\|$. The

solution to this term is of the form

$$\hat{\beta} \;=\; \left(K_Q^{-1} K_{QR}\right) \alpha. \tag{5.29}$$

Equation 5.29 appears in the solution of the *Reduced Subset Problem*, described by [Schölkopf et al. (1999)], which is to find a small set of examples Q and coefficients β to represent a point α in span of a larger set R. However, we use Equation 5.29 to project onto a larger set. Whereas the *Reduced Subset Problem* assumes that $|Q| \ll |R|$ and $Q \subset R$, we assume $|Q| \gg |R|$ and $R \subset Q$. Concatenating the two projections together results in

$$\hat{\beta} \;=\; K_Q^{-1} K_{QR} K_R^{-1} \mathbf{k}_R(q) = K_{RQ}^{+} \mathbf{k}_R(q). \tag{5.30}$$

Equation 5.30 has the same form as the subset projection of Equation 5.13 with $W = 0$, and thus can be computed in the same fashion.

If the approximate distance $\hat{d}$ uses a subset where $R \subseteq Q$ and a subset projection with $W = 0$, then for any $q \in \mathcal{Q}$ and $q_i \in R$, $\hat{d}(q, q_i) = d(q, q_i)$. Although the approximation $\hat{d}$ of the distance is accurate, the value $\hat{d}(q, q_i)$ can be quite large. Without further assumptions, there are no general inequalities between distances $\hat{d}(q, q_j)$ and $\hat{d}(q, q_i)$, with $q_i \in R$, $q_j \in Q \setminus R$. Given the inequality $d(q, q_i) > d(q, q_j)$ on actual distances, the inequality $\hat{d}(q, q_i) > \hat{d}(q, q_j)$ on approximate distances would be desirable, but there are examples where the inequality on approximate distances does not hold.

5.9 Related Works

Our sparse distance approximation problem is related to the out-of-sample extension to the dimensionality reduction problem [Yang and Jin (2006); Bengio et al. (2003)]. For the dimensionality reduction problem, a data set $X = \{x_1, \ldots, x_n\}$ and a metric

$d(\cdot,\cdot)$ are given. The goal is to find a set of points $Y = \{y_1, \ldots, y_n\} \in \mathbb{R}^m$, such that each y_i "represents" its counterpart x_i. This "representation" is defined by either local or global constraints with regard to X and Y. The goal of the out-of-sample extension to dimensionality reduction is to compute a new point y given a real-time point x, without recomputing the mapping of X to Y.

The out-of-sample extension is compatible with algorithms that solve variants of the dimensionality reduction problem such as Multi-Dimensional Scaling [Cox and Cox (2000)], Spectral Clustering [Weiss (1999)], Laplacian Eigenmaps [Belkin and Niyogi (2003)], Isomaps [Tenenbaum et al. (2000)], and Locally Linear Embedding [Roweis and Saul (2000)]. These algorithms construct a (problem specific) kernel function $k_D(\cdot,\cdot)$ dependent on the sample data D and the distance function $d(\cdot,\cdot)$ [Bengio et al. (2003)]. With k_D, a Gram matrix K of the sample data D is computed. Given an out-of-sample element x, its corresponding value y is computed from a kernel projection onto the eigenvectors of K using k_D [Schölkopf and Smola (2001)].

Nyström Method

Our proposed solution to the distance approximation problem is also comparable in performance to solutions which employ matrix approximation techniques such as the Nyström method. The Nyström method has been used to speed up the computation of kernel machines [Williams and Seeger (2001)] and has been used for improved performance in applications such as clustering [Chitta et al. (2011)] and manifold learning [Talwalkar et al. (2008)]. Given an $n \times n$ positive definite matrix G and a parameter $m \ll n$, one can use the Nyström method to produce an $n \times n$ matrix $\tilde{G}^+$ of rank m that approximates G^{-1}. The runtime complexity of producing $\tilde{G}^+$ with the Nyström method is $O(m^2 n)$. This is asymptotically more computationally efficient than the $\Theta(n^3)$ runtime complexity of a standard matrix inversion algorithm.

Kernel Projection Approximation Performance		
Method	Offline Computation	Online Computation
Nyström	$\theta(m^2n + mne)$	$\theta(ne + mn)$
Kernel Semi-least Squares	$\theta(mn^2 + mne)$	$\theta(me + mn)$

Table 5.1: Kernel Semi-least Squares Runtime Complexity
The variable e is the runtime complexity of the kernel function k and n is the size of the dataset Q. For the Nyström method, m is the rank of the approximate Gram matrix inverse $\tilde{K}^+$. For the Kernel Semi-least Squares method, m is the size of the subset $R \subseteq Q$. Typically $m \ll n$.

The input to this calculation is an $n \times n$ matrix G and $m \ll n$ columns sampled from G, represented as an $n \times m$ matrix $G_{n,m}$. The $m \times m$ matrix $G_{m,m}$ consists of the intersection of these m columns with the corresponding m rows of G. The matrix $\tilde{G} \approx G_{n,m}G^+_{m,m}G^*_{n,m}$ is an approximation of G. Analogously, the matrix $\tilde{G}^+$ is an approximation of G^{-1}, and can be computed from $G_{n,m}$ and the singular value decomposition of $G_{m,m} = U_m\Sigma_m U^*_m$. The approximate eigenvalues $\tilde{\Sigma}$ and eigenvectors $\tilde{U}$ of G are $\tilde{\Sigma} = \left(\frac{n}{m}\right)\Sigma_m$ and $\tilde{U} = \sqrt{\frac{m}{n}}G_{n,m}U_m\Sigma^+_m$, respectively. From $\tilde{\Sigma}$ and $\tilde{U}$, the approximation $\tilde{G}^+ = \tilde{U}_m\tilde{\Sigma}^+_m\tilde{U}^*_m$ of G^{-1} is computed.

The Nyström method can be used to create the approximation, $\hat{\beta} = \tilde{K}^+\mathbf{k}_Q(q)$, of the standard kernel projection, $\beta = K^{-1}\mathbf{k}_Q(q)$, where K is the Gram matrix with respect to the kernel function k and the training set Q. The complexity analysis of the kernel projection approximation derived from the Nyström and Kernel Semi-least Squares methods can be seen in Table 1. For the analysis of the offline computations, we assume the kernel function k is only computed on entries in the Gram matrix K. The Nyström method provides computational savings in the offline computation of K^{-1} whereas the Kernel Semi-least Squares method provides computational savings in both the offline computation of K^+_{RQ} and the online computation of the kernel empircal map $\mathbf{k}_R(q)$ of the subset $R \subseteq Q$.

5.10 Experiments

We tested the Kernel Semi-least Squares method on the Sheffield Face Database [Graham and Allinson (1998)]. The face database consists of 564 images of 20 individuals, covering a mixed range of race, sex, and age. The images of the faces were in range of poses including profile and frontal views. There were an average of 28 images per individual. Each picture is approximately 220 $\times$ 220 greyscale pixels represented with 256-bits. Example pictures of the individuals in different poses can be seen in Figure 5·3.

Figure 5·3: Example images from two individuals in the Sheffield Database

We used a type of Hamming distance, $h(\cdot, \cdot)$, as the target distance. To compute this distance, two input images are first converted from greyscale to binary. The greyscale value at every position in the image is set to 1 if it is above a predetermined threshold, and it is set to 0 otherwise. The Hamming distance is computed by counting the number of positions where the two converted binary images have different values. Each individual has a unique threshold, determined by aver-

aging all the greyscale values of all the individual's images in the database. This experiment represents the envisioned application of the Kernel Semi-least Squares method: the distances are expensive to compute, but, as our experiment shows, they can be isometrically-embedded into a low-dimensional vector space.

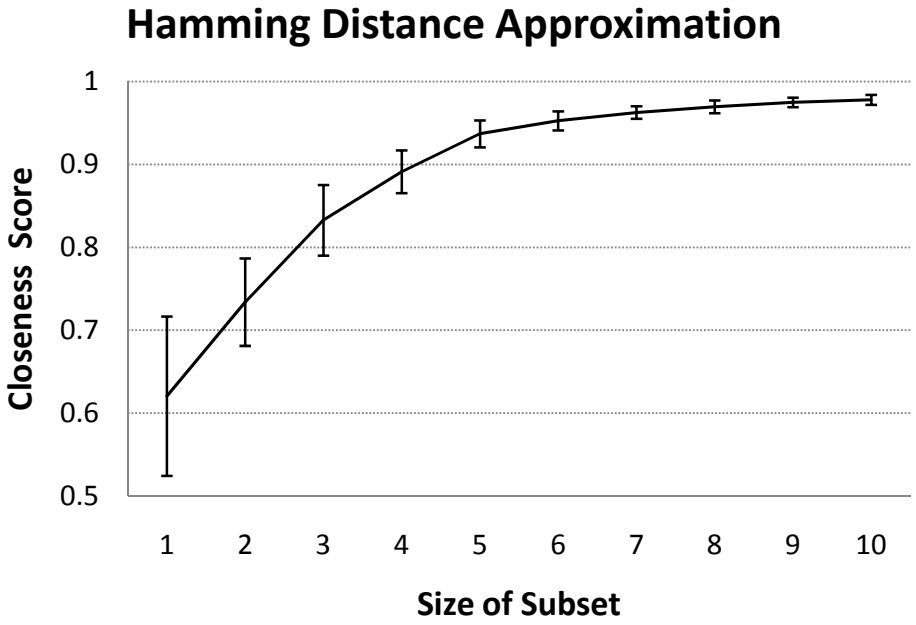

Figure 5·4: The performance of our Kernel Semi-least Squares method for distance approximation for each subset size. The closeness score was determined using leave-one-out cross-validation over 20 individuals. There was an average of 28 images per individual. The error bars represent the standard deviation of the closeness score.

For each individual, we tested the accuracy of the approximate Hamming distance $\hat{h}(\cdot,\cdot)$ using leave-one-out cross-validation. Each image q from the individual's set of images Q was removed in turn and from the remaining group, Q/q, the optimal subsets R_m of sizes $m = 1 \ldots 10$ were computed. Each R_m minimized the cost function of Equation 5.21 over all subsets of size m, with the set of candidate subsets being of the form $\mathcal{R} = \{R : R \subset Q, |R| = m\}$. For each subset R_m of size m, we constructed an approximate Hamming distance $\hat{h}_m(\cdot,\cdot)$. We used the kernel of Equation 6.2, with d being the target Hamming distance and with $g(\cdot)$ set to a constant function. We

chose this kernel because it can be seen as a variant of an intersection kernel, which we have seen has good discriminatory properties. The projection matrix W of the subset projection used in the distance approximation (Equation 5.14) was set to 0. The *difference score* for each image q removed from the set of images Q, using the best subset of size m, was

$$D(m,q,Q) \;=\; \frac{1}{|Q/q|}\sum_{q'\in Q/q}\frac{\left|\hat{h}_m(q,q')-h(q,q')\right|}{h(q,q')}. \tag{5.31}$$

The *closeness score* for each subset size m was computed from the average of the difference score over all individuals $\mathcal{I}$, with

$$S(m) \;=\; 1-\frac{1}{|\mathcal{I}|}\sum_{Q\in\mathcal{I}}\frac{1}{|Q|}\sum_{q\in Q}D(m,q,Q). \tag{5.32}$$

We computed closeness score for subset sizes 1 to 10. Our results show that a small subset can be used to approximate the Hamming distance with a high degree of accuracy, as seen in Figure 5·4.

Nyström Method Comparison

We also compared the performances of two solutions to our posed problem of distance approximation. The first solution applied the Kernel Semi-least Squares method as described in this chapter. The second solution is identical to the "Kernel Semi-least Squares solution" except the Nyström method was used to approximate the kernel projection, as described in section 5.9. For the experiments, we used images of faces from the CMU Pose, Illumination, and Expression (PIE) Database [Sim et al. (2002)]. The database consists of color pictures of faces of individals under different llumination conditions, poses and expressions. We selected 20 individuals randomly from this dataset. For each individual selected, a dataset Q was constructed, consisting of 78 manually cropped 150×250 color images of the person's face under different

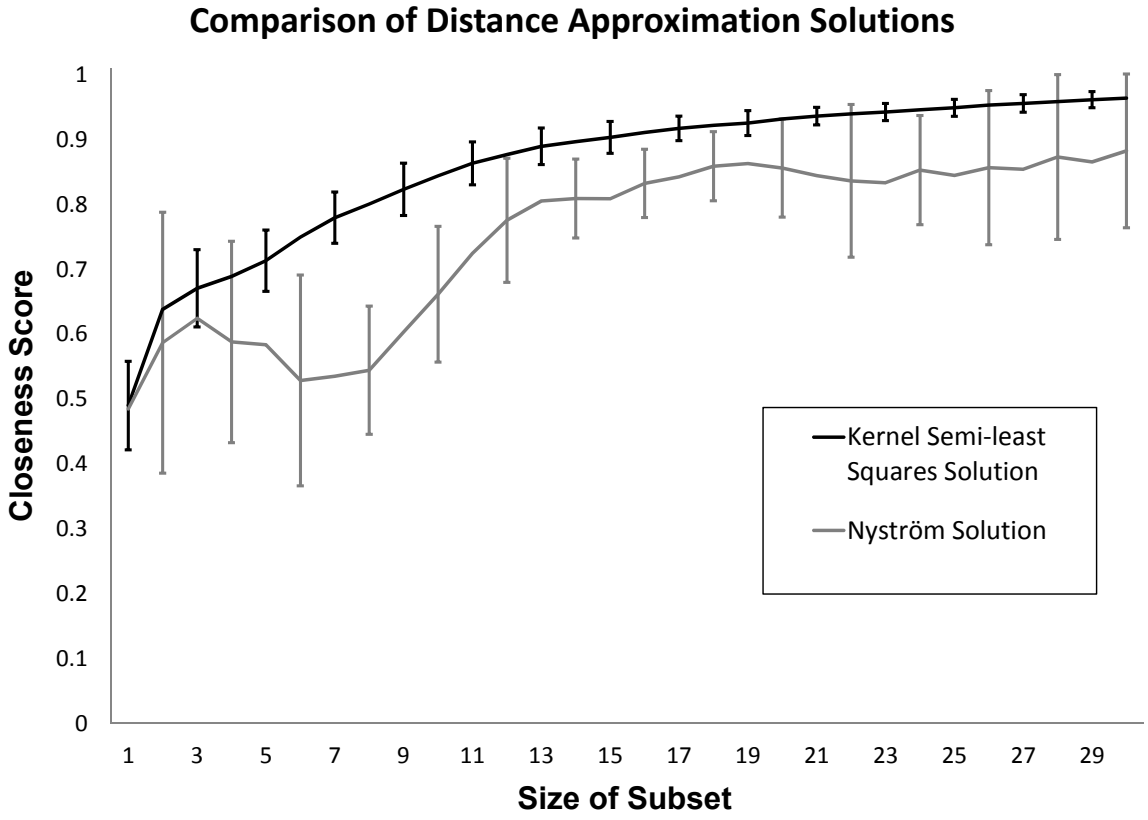

Figure 5·5: The performance of our Kernel Semi-least Squares method in relation to the Nyström method in the task of distance approximation. The closeness score was computed using leave-one-out cross validation over 20 individuals. There were 78 images for each individual. The error bars represent the standard deviation of the closeness score.

illumination conditions and poses.

For each individual dataset Q, we tested the "Kernel Semi-least Squares solution" and the "Nyström solution" on the hamming distance $h(\cdot, \cdot)$ using leave-one-out cross validation in the same manner as the previous experiment. For each individual dataset, the optimal subsets of sizes $n = 1 \ldots 30$ were selected using the criteria of section 5.7.

The results were measured and aggregated over the individuals using the distance $D(m, q, Q)$ and closeness $S(m)$ scores described earlier in this section. The results, as shown in Figure 5·5, indicate the Kernel Semi-least Squares method produced a more

accurate and precise approximation of the target distance than the Nyström method.

5.11 Discussion

We presented a kernel based solution to the sparse distance approximation problem, where the given distance metric d is too computationally expensive to compute exhaustively. Our method uses the subset projection to map an observation to the span of a training set in the kernel feature space. The derivation of the subset projection is derived by extending Rao and Mitra's Semi-least Squares problem with kernel methods.

The kernel projection requires the computation of $d(x, x_i)$ for each $x_i \in X$. In practice, this distance computation can be prohibitively expensive. Our posed problem takes this into consideration, with the aim to have less than n computations of $d(\cdot, \cdot)$. The goal of our formulation is to compute the distances from a real-time element to each element in the training set, instead of a general mapping to a low dimensional Euclidean space.

A benefit of our method is its simplicity. The distance approximation method can be described with two equations (Equations 5.14 and 5.15). The pre-computing requirements are light, consisting of evaluation of the kernel function and matrix (pseudo)inverses.

Future work consists of developing a method for training the arbitrary projection matrix W used in the subset projection. It is an open question how to compute the optimal subset R_o efficiently. We provide several optimization criteria in Equations 5.21, 5.24 and 5.27, which can be computed via enumeration of all possible subsets of size m. The more general question of how closely the coefficient vector β in Equation 5.11 is approximated by $\hat{\beta}$ in Equation 5.14 remains open.

Chapter 6

Camera Mouse

6.1 Introduction

Millions of people worldwide are affected by neurological disorders that cause communication barriers. If individuals with severe traumatic brain injuries, strokes, multiple sclerosis, or cerebral palsy are quadriplegic and nonverbal, they cannot use the computer with a standard keyboard and mouse, or a voice recognition system, as a communication tool. Among individuals with these severe impairments, the Camera Mouse has been established as an assistive communication tool in recent years [Betke (2009)]. Individuals, who can control their head movement, even if the movement range is very small, use systems such as the Camera Mouse as a mouse-replacement interface. The Camera Mouse tracks head movements with a webcam and thereby enables a computer user to control the movement of the mouse pointer [Betke et al. (2002)]. The Camera Mouse tracks a small feature on a user's face, such as a nostril or eyebrow corner. The location of the feature in the camera frame is transformed into the position of the mouse pointer on the screen (Figure 6·1).

The most recent version of the Camera Mouse uses an optical flow approach for tracking [Lucas and Kanade (1981)]. Optical-flow trackers estimate the location of a feature to be tracked by matching the image patch estimated to contain the feature in the previous image with the locally best-matching patch in the current image. Optical-flow trackers are known to incur "feature drift" [Bourel et al. (2000)]. The tracked location may slowly drift away from the initially-selected feature, for which

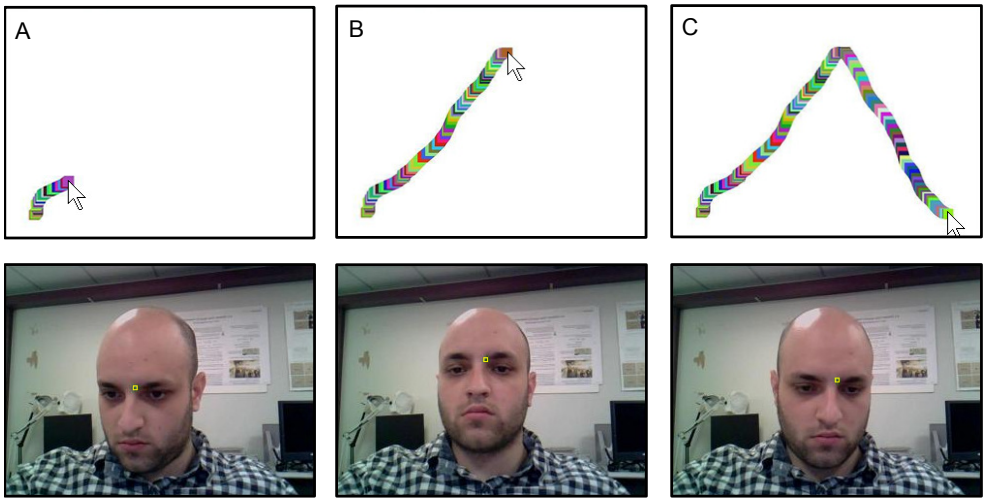

Figure 6·1: Mouse replacement systems enable the user to control the mouse pointer using head movements captured by a webcam. Here, the user is drawing a line with a painting program by moving his head. The feature being tracked is a 10×10-pixel image patch on the subject's left eyebrow. The subject moved his head from his lower left (A), upward (B), and then to his lower right (C). The image coordinates of the feature were translated into screen coordinates for the mouse pointer by a linear mapping.

no record is kept. Camera Mouse users may experience a slow drift of the tracked feature along the nose or eyebrow of the user. Feature loss can also occur when a spastic user makes a rapid involuntary head motion.

To address the problems of optical-flow tracking, we introduce the Kernel-Subset-Tracker. The Kernel-Subset-Tracker uses an exemplar-based approach to track the user's head. A training set of representative sample images of the user's face (or regions of the face) are collected at the beginning of the computer session. After the setup phase, these images are used to create template images for positional tracking. Our approach is based on kernel projections [Epstein and Betke (2009, 2012)], a technique from classification theory.

We here report a significant improvement of the communication bandwidth of test subjects when the Camera Mouse is augmented with the Kernel-Subset-

TRACKER. We refer to this system as the *Augmented Camera Mouse* to distinguish it from the standard Camera Mouse. The Augmented Camera Mouse tracks facial features accurately, without any notable drift, even when subjects move their heads quickly or through extreme orientations, and in the presence of background clutter. We also report that the Augmented Camera Mouse successfully tracked the eyebrow of a user with severe movement impairments. The user was thus able to generate mouse-click events by raising his eyebrow.

6.2 Related Work

Assistive technology offers many hardware devices for people with motion impairments, but very few video-based mouse-replacement systems. A database of information about assistive technology, [ABLEDATA (2010)], lists more than 36,000 products for users with disabilities. The database category "mouse emulation programs" has only 58 entries, and most of these describe education software to be used with physical switches. Only two systems listed offer camera-based mouse-pointer control: the Camera Mouse and the Quick Glance 3 ™ mouse emulator system by EyeTech Digital Systems. [Quick Glance 3 (2010)] illuminates the user's face with infrared lighting and tracks his or her pupils using infrared-sensitive cameras. Other infrared-based commercial mouse-replacement systems are the [SmartNAV (2010)] system by NaturalPoint, which follows a reflective dots attached to the user's head, and the [RED Eye Tracking System (2010)] by SensoMotoric Instruments. Another SensoMotoric product, the [iView X HED (2010)], is a head-mounted system for eye tracking. The [QualiEye (2010)] program by Qualilife is a camera-based mouse-replacement system that tracks a user's face using a webcam.

Unfortunately, commercial hardware solutions are often prohibitively expensive for many people with disabilities and their caregivers [Loewenich and Maire (2007)].

The most expensive commercial products are infrared-based eye-trackers that offer a high resolution in estimating gaze direction. Users,however, find it easier to control a mouse pointer with head motions than with their gaze [Bates and Istance (2003)] (in the latter case, users must look at the location of the mouse pointer while in the former case, they may look elsewhere, e.g. to plan their next move). Fortunately, there are a number of free mouse-emulation systems for users with motion impairments.

The Camera Mouse was the first camera-based mouse-replacement interface that was freely available to users with motion impairments [Gips et al. (2000)], for example, to children with cerebral palsy. In the past decade, a number of other systems have been developed and tested successfully with people with motion impairments. The mouse-emulation system *Nouse*, for example, uses two web cameras to track the 3D position of the nose of the user and was tested with 15 users with motion impairments [Gorodnichy et al. (2007)]. Another 3D approach was proposed by [Tu et al. (2007)], which tracked one subject's face using a 3D model with 12 facial motion parameters. Based on the experiments with users with motion disabilities, [Gorodnichy et al. (2007)] pointed out that the smoothness and range of the users' head movements are often overestimated by developers of camera-based interfaces.

[Kjeldsen (2006)] focused on the problem of non-smooth head movements. He created the HeadTracking Pointer, a mouse-replacement system that converts head movement to pointer movement with a sigmoidal transfer function. The function adapts the transfer rate based on the predicted mouse pointer destination and thus yields smooth mouse pointer movement. A preliminary camera-based mouse-replacement system, using traditional template matching techniques, was created by [Kim and Ryu (2006)]. [Palleja et al. (2008)] described a mouse-replacement system that tracks the head and detects blinks and mouth movements. [Kjeldsen (2006)] and [Kim and Ryu (2006)] mentioned plans to test the proposed interfaces with users with motion

impairments.

[Manresa-Yee et al. (2008)] tested an interface developed by [Varona et al. (2008)] with 10 users with movement disabilities. Interface tracks multiple features on a subject's face, such as the nose, eyes, and mouth. The tracker can recover from tracking failures of individual features through support from other features. Tracking was accomplished using intensity gradients in the video frames. Using the same interface, eight users with movement disabilities reportedly controlled the temperature and lighting of a room [Ponsa et al. (2009)].

Another camera-based mouse-pointer manipulation system was designed by the authors [Loewenich and Maire (2007)]. This system uses a boosted cascade of classifiers to detect a user's face in the video. During tracking, a collection of features are tracked using optical flow. This system was tested with 10 volunteers without movement disabilities.

It will be exciting to see how the computer vision techniques discussed above will improve the accuracy of facial feature tracking so that camera-based mouse-replacement systems can be successful tools for the larger community of people with movement disabilities. At this time, unfortunately many individuals with severe movement disabilities, who use mouse-replacement systems, gain only limited control of the mouse pointer. This is due to the difficulties many users have in positioning the mouse over traditional target areas such as buttons or web links.

Research efforts have been made to adjust application software so that it can be used successfully with a mouse-replacement system. Examples are the WebMediator, a program that alters the display of a web page so that the fonts of links become larger [Waber et al. (2006)] and the CameraCanvas, an image editing tool for users with severe motion impairments [Kim et al. (2008)]. Another example is the Hierarchical Adaptive Interface Layout (HAIL) by [Magee and Betke (2010)], which

is a set of specifications for the design of user interface applications, such as a web browser and a Twitter client, that adapt to the user. In HAIL applications, all of the interactive components take place on configurable toolbars along the edge of the screen.

[Hwang et al. (2004)] reported that some users with impairments pause the pointer more often and require up to five times more submovements to complete the same task than users without impairments. [Wobbrock and Gajos (2008)] focused on the difficulty that people with motion impairments have in positioning the mouse pointer within a confined area to execute a click command. They introduced "goal posts" which are circular graphical boundaries that trigger application actions when crossed with the mouse pointer. [Findlater et al. (2010)] used this idea to create "area cursors" that use goal-crossing and magnification to ease selection of closely positioned interface targets. [Betke et al. (2006)] proposed to discretize user-defined pointer-movement gestures in order to extract "pivot points," i.e., screen regions that the pointer travels to and dwells in. Related mechanisms are "gravity wells" that draw the mouse pointer into a target on the screen once it is in proximity of the target [Biswas and Samanta (2007)] and "steady clicks," a tool that reduces button-selection errors by freezing the pointer during mouse clicks and by suppressing clicks made while the mouse is moving at a high speed [Trewin et al. (2006)].

The Camera Mouse system may be the most-used freely-available camera-based mouse-replacement system to date. It has been downloaded 500,000 times as of the date of this printing and is popular with users. Our new tracker, the Kernel-Subset-Tracker, is designed to support current Camera Mouse users and also empower new users, who could not use the Camera Mouse previously due to frequent feature loss. We incorporated the proposed Kernel-Subset-Tracker into the original Camera Mouse software. The new tracker can be toggled on and off to suit

the needs of the user.

6.3 The Kernel-Subset-Tracker

The KERNEL-SUBSET-TRACKER is an exemplar-based tracking algorithm that uses a representative training set to model the objects to be tracked. It requires a training phase at the beginning of the interaction session. In the training phase, a a set of object images is collected as a training set. For face tracking, the training set consists of images of size 100×100 of the face at different orientations of the head relative to the camera. The training set is used to identify the object to be tracked in successive image frames during human-computer interaction. At time t, the KERNEL-SUBSET-TRACKER determines a dissimilarity score, distance d_i, of the current object at position p, to each training image q_i in the training set $Q = \{q_1, q_2, \ldots, q_n\}$. From such distances, a positional template is created and used to find the next position p' of the object in the video frame.

```
1: function KERNEL-SUBSET-TRACKER(t, p)
2:     I = GetVideoFrame(t)
3:     q = GetRealTimeObs(I, p)
4:     for all n training images q_i in Q do
5:         d_i = f(q, q_i)
6:     a = CreateTemplate(Q, d)
7:     p' = PositionSearch(v, p, a)
8:     Output(qt, p', d, a)
9:     KERNEL-SUBSET-TRACKER(t + 1, p')
```

In the KERNEL-SUBSET-TRACKER, see pseudocode above, function GETVIDEO-FRAME returns the complete image frame at the current time t. Function GETREAL-TIMEOBS crops a subimage located at the current position p from the current video frame I. This subimage is the real-time observation q. Function f returns a distance measure between the real-time observation and each training image q_i of the training set Q. For many distance measures, evaluating f exhaustively becomes untenable for

current computers if the distance measure uses every pixel in the input images. In Section 6.4, we describe a method to approximate the distance measure with a kernel (see Section 6.4).

The positional template a is computed by function CreateTemplate, which takes as inputs the distances d and the training set Q. Function PositionSearch computes the optimal local alignment p' of template a, given the current video frame I and the previous position p. Eight subimages are cropped from the current video frame I from windows centered at position p and each of its eight neighbors $p + (-1,-1), p + (0,-1), p + (1,-1), p + (1,0), p + (1,1), \ldots$. The first estimate $\hat{p}'$ of the position is equal to the center position of the subimage that best matches a. The same distance measure used by function f is also used in the PositionSearch method to evaluate the eight alignment candidates. This process is repeated by considering the eight neighbors of $\hat{p}'$. Hill climbing proceeds until none of the neighboring subimages can provide a better alignment or until a fixed number of iterations hasoccurred. PositionSearch then returns the locally best estimate p'. The Output of the Kernel-Subset-Tracker for each frame is the 2D position of the tracked object, the distances d_i of the training images and the positional template a of the tracked object.

6.4 Distance Approximation with Kernels

The most computationally intensive component of the Kernel-Subset Tracker is the repeated calls to the distance method $f(\cdot,\cdot)$ for each training image q_i in the training set Q. We describe how to use kernel methods from machine learning [Schölkopf and Smola (2001)] to approximate the distance function quickly.

Distance functions such as $f(\cdot,\cdot)$ define metric spaces and likewise inner product functions $\langle\cdot,\cdot\rangle$ define vector spaces. The most common inner product is the one for

Euclidean spaces,

$$\langle (x_1, \ldots, x_n), (y_1, \ldots, y_n) \rangle \quad = \quad \sum_{i=1}^{n} x_i y_i.$$

Another example of an inner products is

$$\langle (x_1, x_2), (y_1, y_2) \rangle \quad = \quad x_1 y_1 + x_2 y_2 + (x_1 + x_2)(y_1 + y_2). \tag{6.1}$$

These inner products are also known as kernels. We use the notation of $k(\cdot, \cdot)$ to describe the kernels. If $k(\cdot, \cdot)$ is semi-positive definite then it is a valid kernel [Schölkopf and Smola (2001)].

The main benefit of using kernels is that they endow distance measures with notions of angles and length and so projections can be used. Given the distance function f, we can create a kernel function $k(\cdot, \cdot)$ whose induced distance is equal to the function f. Thus the function f can be isometrically embedded in the vector space implied by the kernel[1]. We define such a kernel function $k(\cdot, \cdot)$ by

$$k(q, q') \quad = \quad h(q) - \frac{1}{2}\big(f(q, q')\big)^2 + h(q'), \tag{6.2}$$

for any arbitrary function $h : \mathcal{Q} \to \mathbb{R}$. In practice, however, it is easer to define the kernel function directly.

Using the subset projection method described by [Epstein and Betke (2009)], we do not need to compute the distance function f between the real-time observation q and every training image q_i. Instead, we can compute a kernel function $\hat{f}$ that represents the distance between a real-time observation q and a small subset of the training images $R \subset Q$, with $R = \{r_1, \ldots, r_m\}$. The results of these inner products can be used to approximate the distances d_i. The pseudocode of KERNEL-SUBSET-TRACKER can be modified to accommodate this subset projection method by replacing lines

[1]This is assuming the distance function is Hilbertian.

4: **for** all n training images q_i in Q **do**

5: $\quad d_i = f(q,q_i)$

by the subset projection functionality:

4: $R = \text{RANDOMSUBSET}(Q, d^{\text{prev}})$

5: **for** all m training images r_j in R **do**

6: $\quad v_j = k(q, r_j)$

7: **for** all n training images q_i in Q **do**

8: $\quad d_i = \hat{f}(q, q_i, v)$

The RANDOMSUBSET method returns a random subset R of the training images Q. The probability that a training image q_i will be chosen for a subset R is inversely proportional to its distance to the previous real-time observation d_i^{prev}. Thus, training images that are similar to the real-time observation of the previous frame have a higher probability to be in subset R. In practice, the distances to a training set Q of size 25 can be approximated using the subset projection method and a small set R of size 5.

6.5 Three Kernels for the Kernel-Subset-Tracker

In this section, we define three image-based kernels used in our experiments. An image-based kernel is a function of two grayscale images that returns a real number representing their inner product. A simple example of an image-based kernel function is one which returns the sum of the pairwise product of the intensity values of the

images. On input images q and q' of size 100×100, this kernel returns

$$k(q, q') = \sum_{x=1}^{100} \sum_{y=1}^{100} q(x, y) * q'(x, y), \tag{6.3}$$

with $q(x, y)$ representing the brightness of image q at position (x, y).

6.5.1 Threshold Kernel

The threshold kernel is the main kernel we used in our experiments (Figure 6·2). This kernel first performs thresholding of a pair of grayscale images according to threshold τ to produce two processed binary observations. It computes the size of the intersection of the "1" pixels of these two processed observations. For simplicity, this number is divided by the number of pixels of the input images to yield an output between 0 and 1 (the division operation has no effect on the performance of the kernel).

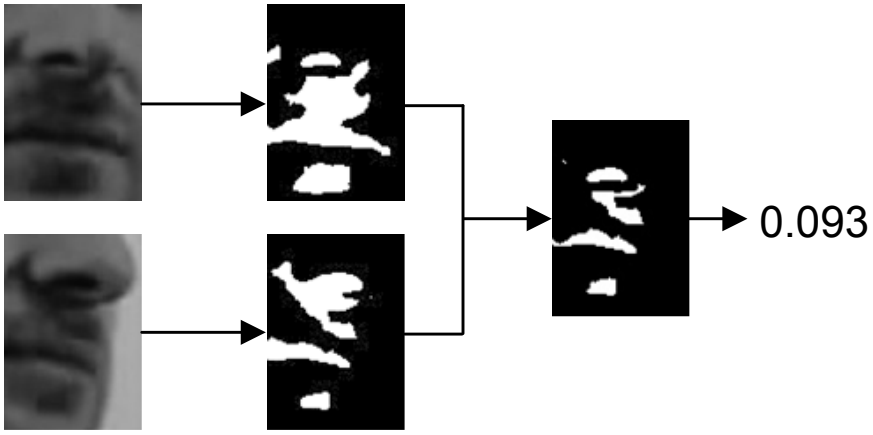

Figure 6·2: An example of the threshold kernel. Two grayscale images are converted to binary images using a set threshold and then combined to a single binary image using the intersection operation. The final output is the percentage of "set" pixels in this combined image.

As we show below, the threshold kernel results in excellent tracking in certain imaging scenarios; however, it is not robust to changes in brightness, contrast, or object scale. This is due to the fixed nature of τ, the thresholding parameter.

6.5.2 Normalized Threshold Kernel

We designed the NORMALIZED THRESHOLD KERNEL to provide a tracking mechanism that is robust to changes in brightness and contrast. This kernel takes as input two grayscale images q and q' and outputs a real number between 0 and 1 (see pseudocode). Each input is converted to a binary image using its mean as the threshold. The size of the intersection of the two binary images is computed. This value is normalized by the number of pixels and returned. This final normalization is a convenience step, having no effect on the performance of the kernel.

```
1: function NormalizedThresholdKernel(q, q')
2:     m =ComputeMean(q)
3:     m' =ComputeMean(q')
4:     c = 0
5:     for x = 1 to width of training images do
6:         for y = 1 to height of training images do
7:             if q(x, y) ≥ m and q'(x, y) ≥ m' then
8:                 c = c + 1
9:     return c / NumPixels(q)
```

The function NORMALIZED THRESHOLD KERNEL is semi-positive definite, and thus a valid kernel. It is invariant to uniform changes in brightness and contrast (Figure 6·3).

6.5.3 Normalized Radial Intensity Kernel

We introduce the NORMALIZED RADIAL INTENSITY KERNEL (NRI) to provide a tracking mechanism that is robust to changes in object scale. The NRI-Kernel computes an inner product on two grayscale images q and q' in the following two part process.

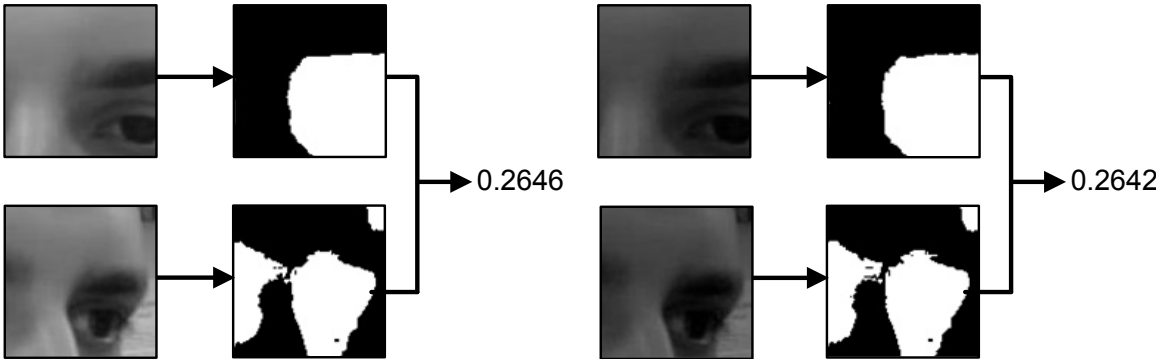

Figure 6·3: NORMALIZED THRESHOLD KERNEL. The images (A) were subjected to the lowering of brightness and contrast (B). Thresholding based on the means of the images results in similar binary images (A and B) and kernel outputs. This is an example of the invariance of the NORMALIZED THRESHOLD KERNEL to uniform changes in brightness and contrast.

The first part converts each grayscale image to an intermediate feature vector, which is a small array of positive real numbers between 0 and 1. Each value of the array represents the summation of intensity values of the image, along a ray from the center of the image proceeding in a specified direction. The array is normalized such that its largest entry is 1.0. An example conversion can be seen in Figure 6·4. We tried a number of different array sizes, including 8 and 16 rays. We found the best performance of the KERNEL-SUBSET-TRACKER when we used 32 directions.

The second part of the NRI-Kernel computes an inner product between the two radial feature vectors v and v' derived from two images. We tried several methods, including the standard sum of pairwise multiplication of the values of the two vectors. However we found the intersection operation resulted in the best tracking results. Thus the NRI-Kernel returns the sum of the pairwise minimum of every pair of values in vectors v and v'. The sum is normalized (divided by 32) so that the output of the NRI-Kernel is between 0 and 1. This normalization is done for ease of comparison, and has no effect on the performance of the kernel.

The NRI-Kernel is invariant to small changes in scale of the object being tracked,

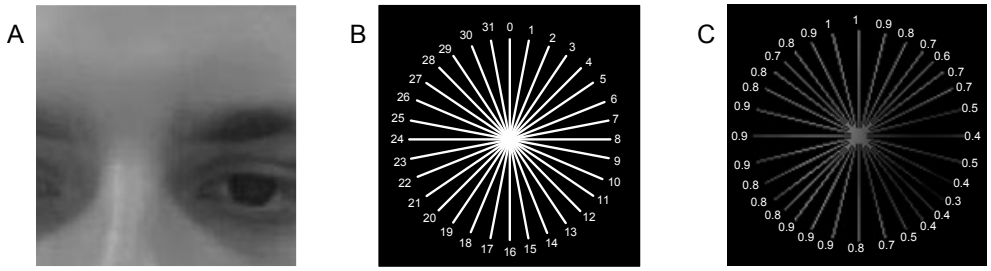

Figure 6·4: A grayscale image (A) is converted into the intermediate feature vector (C) used by the NORMALIZED RADIAL INTENSITY KERNEL. Each number in (C) is an entry of the feature vector, which is created by summing up the intensity values from the center point in the directions shown in image B. The result is a feature vector of 32 positive numbers representing the relative intensity of each radial direction, normalized to be between 0 and 1, as shown in (C), rounded to one significant digit.

since they would not affect the relative intensity values along the radial directions. The normalization operation makes the kernel also invariant to changes of brightness and contrast. Some sample inputs demonstrating this invariance can be seen in Figure 6·5.

6.6 Positional Template Creation

In this section we describe the positional template function CREATEBINARYTEMPLATE we used in the KERNEL-SUBSET-TRACKER in conjunction with both the Threshold Kernel and the Normalized Threshold Kernel. In the CREATEBINARYTEMPLATE function, the positional template a is constructed from the observation set Q, where the contribution of each individual q_i to the output is inversely proportional to its distance d_i to the real-time observation q. Given are the distances d_i of the current frame subimage and the threshold of the Threshold Kernel τ.

The binary image template output a is created by iterating through every pixel position of the training images and setting a temporary value δ to 0. If the grayscale value of training image q_i is greater than threshold τ at the current position index

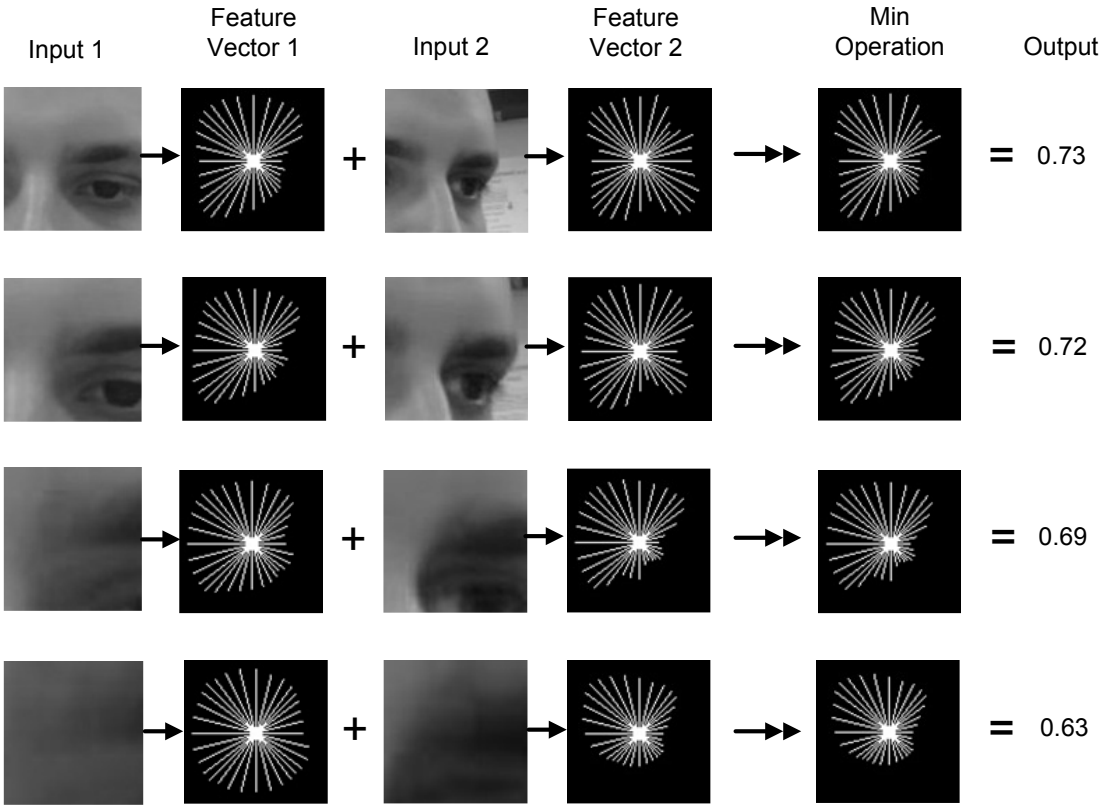

Figure 6·5: The operations of the Normalized Radial Intensity Kernel. Each row shows the two grayscale input images at increasing scales. The NRI Kernel converts each image into an feature vector of size 32, where each value represents the sum of the pixels in a particular direction, starting from the center position. The feature vectors are shown with the lengths of rays representing the magnitude of each value. The arrays are combined into a third feature vector using the minimum operation. The output is the magnitude of this feature vector normalized to be between 0 and 1. The similarity of the outputs exemplifies how the NRI Kernel successfully handles local changes in scale.

$(x_{\text{pos}}, y_{\text{pos}})$, then it will "vote" for a 1 pixel by adding weight $1/d_i$ to δ. Similarly $1/d_i$ will be subtracted from δ if its intensity is below threshold τ. The contribution of each training sample q_i to the construction of a is proportional to $1/d_i$. After all training images have voted, the output a at position $(x_{\text{pos}}, y_{\text{pos}})$ will have intensity 1 if $\delta \geq 0$, otherwise 0.

```
1: function CreateBinaryTemplate(Q, τ, d)
2:     for x_pos = 1 to width of training images do
3:         for y_pos = 1 to height of training images do
4:             δ = 0
5:             for i = 1 to n do
6:                 if q_i(x_pos, y_pos) ≥ τ then
7:                     δ = δ + 1/d_i
8:                 else
9:                     δ = δ − 1/d_i
10:            if δ ≥ 0 then a(x_pos, y_pos) = 1 else 0
11:    return a
```

This binary image is then used by the Kernel-Subset-Tracker algorithm in a local search to find the new position of the object in the frame. This action is performed in the PositionSearch function of the Kernel-Subset-Tracker. At each position in the local search, a grayscale image is cropped from the current video frame. This image is thresholded into a binary image using the threshold τ of the kernel. All of the binary images of the neighboring positions are compared against the template and the current tracking position is changed to that of the closest matching neighboring binary image. This process is repeated until a local maximum is reached.

6.7 Augmenting the Camera Mouse with the Kernel-Subset-Tracker

In the Augmented Camera Mouse, the user can configure the Kernel-Subset-Tracker by selecting a kernel to use, the size the training set, and the size of

the subset projection. During the training phase, Augmented Camera Mouse populates the training set by obtaining a series of pictures of the user's head in different positions. To guide the user in making head movements that yield effective training images, the Augmented Camera Mouse asks the user to perform a simple target-reaching task. In this training phase, the user's motion is tracked with optical flow for bootstrapping. The target-reaching task requires users to move the mouse pointer with their head over a set of blocks on the screen as shown in Figure 6·6. When the pointer enters a block, a subimage of the user's face, which is a 100×100 window around the currently tracked position, is stored as a training image. The number of blocks n^2 (e.g., $n = 2, 3$, or 4), and the size of blocks are configurable. The training phase lasts only a few seconds – as long as it takes the user to move his or her head into the n^2 positions. Retraining is required if the conditions during the computer session change significantly (e.g., the lighting changes or the user starts wearing glasses).

The Augmented Camera Mouse uses both the original optical flow tracking algorithm and the Kernel-Subset-Tracker. At each frame, the old position of the facial feature is updated. The optical flow algorithm first computes an estimate of the position using a 10×10 square patch around the previous position. The Kernel-Subset-Tracker then crops a square window of length 100 pixels around this estimate. The Kernel-Subset-Tracker refines the estimate of the position for the next frame, using the hill climbing PositionSearch algorithm (Section 6.3).

6.8 Experiments with Subjects without Motor Impairments

6.8.1 Participants

We worked with 19 subjects (16 males, 3 females, 20-40 years of age). The subjects did not have motion disabilities.

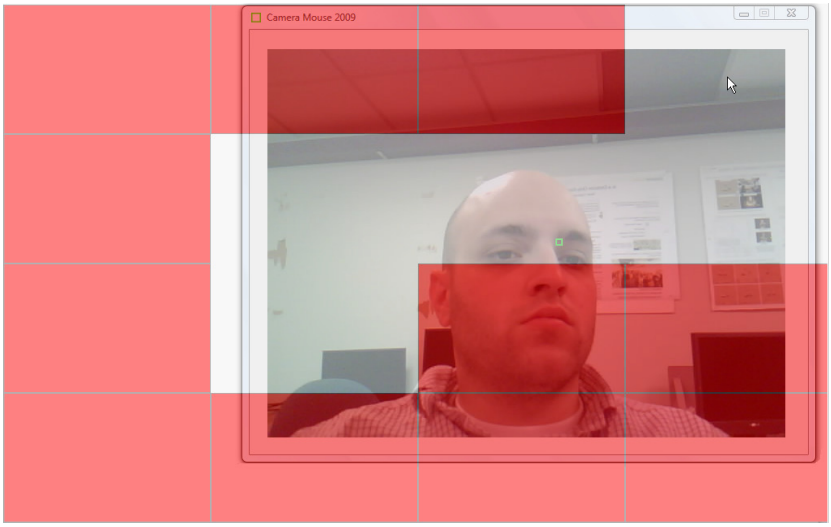

Figure 6·6: Target-reaching task during the real-time image-collecting training phase of the Augmented Camera Mouse. Optical flow is used for tracking as a bootstrapping technique. The screen initially shows the overlay of n^2 red blocks (here 16) that the user is asked to reach with the mouse pointer. When the pointer enters a screen block, the Augmented Camera Mouse obtains a 100×100 subimage of the user's head (centered around the tracked feature) and adds it to the training set. The red overlay disappears to indicate that the screen region has been reached successfully (here, five blocks have been reached and five training images have been obtained).

6.8.2 Apparatus

A Logitech Quickcam Pro 4000, which captures images at a frame rate of 30 Hz, was used as the video capture device. The KERNEL-SUBSET-TRACKER software package was implemented in C++. The experiments were conducted with a laptop with 4 GB of RAM and Intel Core Duo 2.1 GHz processors.

6.8.3 Test Software

We developed test software that encourages subjects to move their head significantly while interacting with the Augmented Camera Mouse interface. Similar to HCI experiments in the past [Akram et al. (2006); International Organization for Standard-

ization (2007)], our test software displays a series of circles that the user targets with the mouse pointer. Each circle appears individually and disappears when the subject moves the mouse pointer to the current circle, triggering the next circle to become visible (Figure 6·7). To induce different types of user motions, we designed three target arrangements that differ in placement, ordering, and sizes of circles.

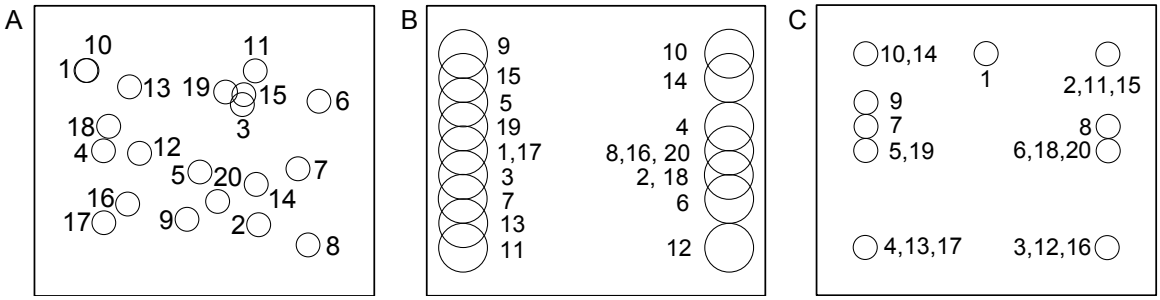

Figure 6·7: The placement, size and ordering of the targets in our experiments. Numbers correspond to the time steps in the experiment.

6.8.4 Test Procedure and Setting

We tested accuracy of the Augmented Camera Mouse with regard to tracking a subject's facial feature during varied head movements (Figure 6·8). The subjects used our testing software in 10 sessions for about thirty minutes, on average. The subjects sat in front of a cluttered background and faced the external monitor that contained the test software that we developed. The test supervisor faced the laptop monitor that contained the Augmented Camera Mouse interface. This interface showed the current tracking positions overlaid on the webcam image (Figure 6·1, bottom). If the Augmented Camera Mouse lost the selected feature, the supervisor would record the event as a tracking failure and reinitialize the mouse pointer by manually resetting the tracking position to the appropriate image feature.

The experiments involved five sessions:

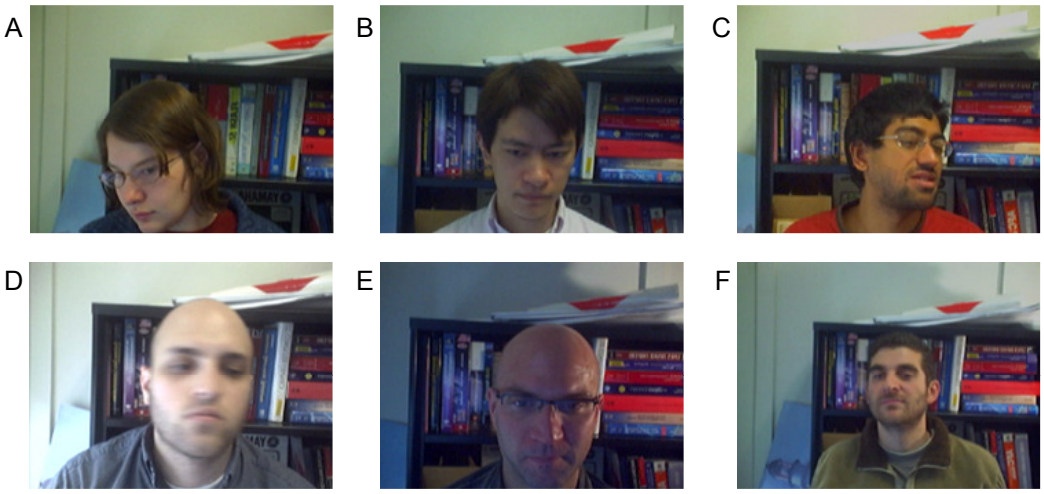

Figure 6·8: Sample images captured by the webcam during the testing phase. The images show different head orientations (A, B, C), rapid motions (D), changed lighting (E), and changed scale (F). All subjects were tested in front of the cluttered bookcase shown in the images.

- **Normal Session**. The subject was instructed to move the mouse pointer to a series of 20 randomly placed targets. This session represents the typical motions and orientations which a Camera Mouse user would encounter in day-to-day operations (Fig. 6·7 A).

- **Hastened Session**. A total of 20 targets were placed alternatively on the left and right side of the screen. The subject was instructed to move the mouse as quick as possible. This session was designed to induce large horizontal motions (Fig. 6·7 B). We chose not to use vertical motions to decrease neck strain in the users.

- **Boundary Session**. This session was designed to have the subject occlude large portions of his or her face due to moving the head in extreme positions. A total of 20 targets were placed along the boundary of the screen (Fig. 6·7 C).

- **Changed Lighting Session**. The subject was instructed to move to the same

target arrangement as those of the normal session. The overhead lights in the room were turned off to create darker lighting condition than that during the setup phase (Fig. 6·7 A).

- **Changed Scale Session**. This session used the same arrangement as the normal session, but with the camera moved two feet away from the subject. This resulted in smaller scaled features (Fig. 6·7 A).

For consistency, the order of the sessions and trackers was fixed for all subjects. We first worked with the Standard Camera Mouse and then the Augmented Camera Mouse. We tested the perfomance of a given tracker in only the sessions that were appropriate for it. We tested the Augmented Camera Mouse with the Threshold Kernel, the Normalized Threshold Kernel, and the Normalized Radial Intensity Kernel, defined in Section 6.5. Using the Augmented Camera Mouse with the Normalized Threshold Kernel in the normal and changed-lighting sessions, we tested the invariance of the kernel to differences in feature illumination. In the changed-scale sessions, we tested the invariance of the Normalized Radial Intensity Kernel to changes in size of the tracked feature. We also compared the performance of the Standard Camera Mouse and Augmented Camera Mouse with the threshold kernel during the normal, hastened, and boundary sessions.

During each session the Augmented Camera Mouse used 25 training images and subset projections of size 5. The limit for the number of steps of the hill climbing algorithm for any video frame was set to 10.

The facial feature tracked was the inner left eyebrow corner. We selected it since it is centered in the face, and not likely to be occluded. From our experience, when the eyebrow was the feature tracked, subjects required less cognitive processing in converting head motions to mouser pointer motions.

6.8.5 Analysis Procedure

To evaluate the tracking accuracy of the Augmented Camera Mouse, we compared computed feature positions against manual "ground-truth" of feature locations. For each session, an image of the webcam was saved once per second. After the session was over, an independent observer used a custom program to mark the location of the facial feature in each image. For each session, the average Euclidean distance between the target locations and the manually marked locations was computed. We use this distance to represent the error of the tracker with regard to the hand-marked ground truth.

We also evaluated the potential of "feature drift," in which a tracked point diverges away from the initially selected feature. The issue of feature drift particularly arises when trackers are used for extended periods of time. The drift measure can be approximated by the increase of the error of a tracker over time. For each subject session, feature drift is determined by the slope of the best linear fit of the error, as computed above, versus time into the session. Feature drift is measured in units of pixels per second.

Between 18 and 64 images were saved per session, with an average of 34 images. The average time to manually mark the ground truth for each subject was 45 minutes.

We evaluated the benefit of the Augmented Camera Mouse with an HCI theoretic performance measure known as the *Index of Performance* [MacKenzie (1989)]. This measure describes the performance of one or many users with a particular device. The Index of Performance is also known as the bandwidth of the device, with units in bits per second. The measure is similar to the performance indices of the electronic communication devices, with larger values signifying better performance.

The Index of Performance can be approximated using Fitts' law [Fitts (1954)]. Fitts' law says that for pointing devices, the average time it takes a user to use a

device to point to a target is linearly related to the level of difficulty of the task. It can be stated succinctly as

$$MT = c_1 + c_2 \times ID, \tag{6.4}$$

where MT represents the (mean) time to reach a target, ID is the index of difficulty of reaching the target, and c_1 and c_2 are constants dependent on the device and the user. Of the many variants of the index of difficulty, we use an information theoretic formulation [MacKenzie (1989, 1992)],

$$ID = \log\left(\frac{D}{W} + 1\right), \tag{6.5}$$

where D is the distance to the target and W is the diameter of the target. The Index of Performance (IP) for a particular user and device is

$$IP = 1/c_2, \tag{6.6}$$

with units of bits per second. We found the Index of Performance experimentally by collecting the behavior of our group of subjects performing a number of *actions* with a particular device. For our purposes the device is the Standard or Augmented Camera Mouse with different kernels. An action represents the task of moving the mouse pointer to a target. A user performing the mouse tracking experiment with one of the target arrangements shown in Figure 6·7 produces 19 actions. Each action is represented by a (*Movement Time*, *Index of Difficulty*) pair, which contains the time to move the mouse from the previous to the new target position and the Index of Difficulty of the task, as described in equation 6.5. The terms W and D are the width and distance between the targets in screen pixels, with ranges of [100, 200] and [128, 976], respectively.

6.8.6 Results

Using the KERNEL-SUBSET-TRACKER with the threshold kernel, the Augmented Camera Mouse achieved a frame rate of 30 fps. The other kernels defined in Section 6.5 are more computationally expensive, but still achieved a frame rate of 30 fps.

We evaluated the tracking accuracy of the Augmented Camera Mouse (Table 6.1). The Augmented Camera Mouse with the threshold kernel during the normal, hastened, and boundary sessions performed with an average Euclidean error distance of 6.1, 7.9, and 7.7 pixel widths, respectively. On average, the width of the eyebrow of the subjects was 63 pixels. The error in localizing the eyebrow corner was therefore only about 1/10 the length of the eyebrow, implying the Augmented Camera Mouse tracked the left eyebrow with a high degree of accuracy.

Table 6.1: Tracking Error. Average and standard deviation of the Euclidean distance in pixels widths between the feature position estimated by the tracker and the ground truth marking. Each session had 19 subjects and on average 34 images. The field of view of the webcam is 640 by 480 pixels. The × symbol marks sessions that were not tested.

Tracker Error (Pixels)					
Tracker	Sessions				
	Normal	Hastened	Boundary	Changed Lighting	Changed Scale
Standard Camera Mouse	9.7 ± 6.2	13 ± 5.8	13 ± 5.9	×	×
Threshold Kernel	6.1 ± 2.7	7.9 ± 2.6	7.7 ± 2.7	×	×
Normalized Threshold Kernel	5.8 ± 2.5	×	×	9.2 ± 7.1	×
Normal. Radial Intensity Kernel	6.5 ± 1.7	×	×	×	5.6 ± 2.2

The pairwise difference in accuracy of the Augmented Camera Mouse with the threshold kernel versus the Standard Camera Mouse was statistically significant. In the random, hastened and boundary sessions, the $(p, t(18))$ results were $(0.004, 0.002)$, $(0.006, 0.003)$, and $(0, 0)$ respectively, based on a t-test with 18 degrees of freedom.

The Augmented Camera Mouse was empirically shown to be very resilient to feature drift (Table 6.2). The average feature drift for all configurations used by the Augmented Camera Mouse was very close to zero, except for the hastened session

with a modest drift of 0.1 pixels per second. The pairwise difference of feature drift of the Augmented Camera Mouse with the threshold kernel versus the Standard Camera Mouse was statistically significant, with $p = 0.0$, $t(18) = 0.0$ in the random and boundary sessions. For the hastened sessions a weak statistical significance was found, with $p = 0.22$ and $t(18) = 0.11$.

Table 6.2: Drift Error The drift metric represents the rate of error increase of a tracker over time. For each subject session, the feature drift is determined by the slope of the best linear fit of the error versus time into the session. The values below represent this feature drift, averaged over 19 subjects. They are in units of pixels per second. The error is determined as for Table 6.1.

Drift Error (Pixels Per Second)					
Tracker	Sessions				
	Normal	Hastened	Boundary	Changed Lighting	Changed Scale
Standard Camera Mouse	0.22 ± 0.29	0.25 ± 0.38	0.37 ± 0.31	×	×
Threshold Kernel	0.0 ± 0.07	0.1 ± 0.24	0.03 ± 0.1	×	×
Normalized Threshold Kernel	-0.02 ± 0.16	×	×	0 ± 0.1	×
Normal. Radial Intensity Kernel	-0.02 ± 0.11	×	×	×	-0.01 ± 0.1

We empirically tested the invariance of the specialized kernels to changes in lighting and scale. The average error of the normalized threshold kernel was comparable to average error of the regular threshold in the normal sessions, in terms of average error. The normalized threshold kernel was shown to be generally invariant to changes in lighting conditions. The average error in the changed lighting session increased by 58% to 9.2 ± 7.1 pixels. The average and variance of the feature drift are equal in the normal and changed lighting sessions with the normalized threshold kernel. Their pairwise difference had $p = 0.82$ and $t(18) = 0.41$, indicating no statistical significance.

The Normalized Radial Intensity Kernel (NRI Kernel) proved to be very effective in tracking the eyebrow at different distances from the camera. The average tracker error of the NRI Kernel for the normal and changed scale sessions decreased from 6.5 pixel widths to 5.6 pixel widths. The increased distance of the users to the

cameras, which results in smaller faces in the captured image, is a likely reason for the decrease. Similar results were achieved for the feature drift of both sessions with the NRI Kernel. A pairwise comparison resulted in $p = 0.69$ and $t(18) = 0.34$. indicating no statistical significance in the difference of drift.

Both the Augmented Camera Mouse and the Standard Camera Mouse had occasional tracking failures. In particular when the subject had extreme motions, we measured the same number of tracking failure losses in the Standard Camera Mouse and with the Augmented Camera Mouse using the threshold kernel. The Standard Camera Mouse had three tracking failures in the hastened sessions. The Augmented Camera Mouse with the threshold kernel had one tracking failure in the normal sessions and two in the hastened sessions. The tests for lighting and scale invariance resulted in a single extra tracking loss in one of the changed lighting sessions.

The Index of Performance of the Augmented Camera Mouse was derived from the inverse slope of the best linear fit of the actions (Figure 6·9). The Index of Performance of the Augmented Camera Mouse was higher than the Standard Camera Mouse in the Normal and Boundary Sessions, e.g., 2.9 bits/s versus 1.4 bits/s (Table 6.3). In both sessions, users were instructed to move naturally. This indicates when the users did not rush with the devices, they performed the tasks quicker with the Augmented Camera Mouse than the Standard Camera Mouse. In the hastened sessions, we instructed users to move as quick as possible, and devices had equal Indices of Performance, due to the rushed motions of the users. Sessions using the Normalized Threshold and Normalized Radial Intensity Kernels had performance measurements lower than the Threshold Kernel, but higher than the Standard Camera Mouse. The changed lighting and scale sessions resulted in slightly lower performance of the Augmented Camera Mouse.

We did not randomize the order of the experiments. During the experiments,

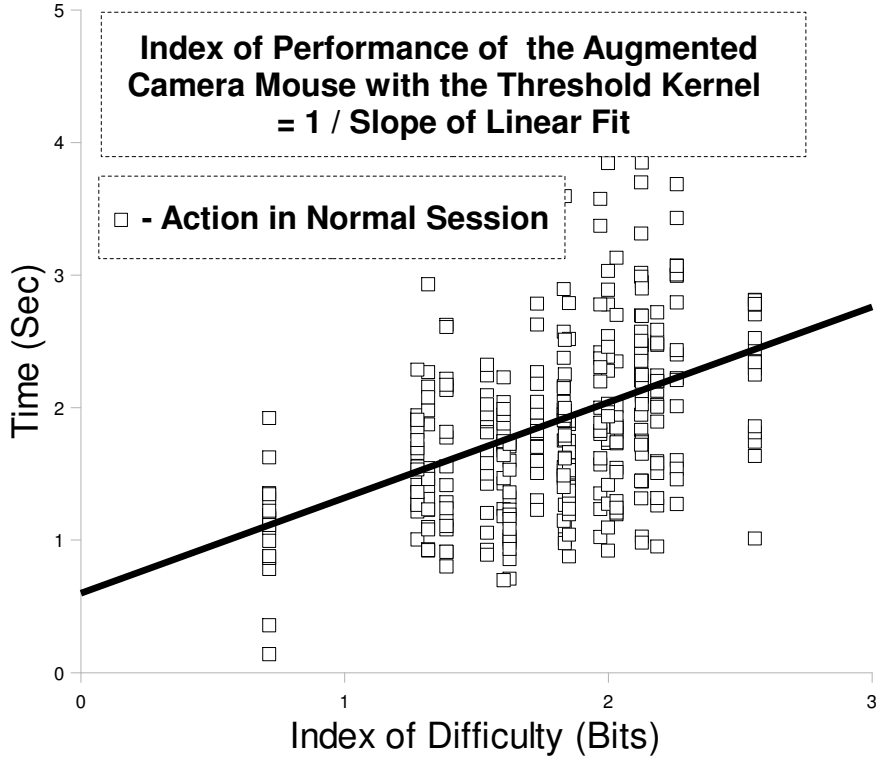

Figure 6·9: Index of Performance of the Augmented Camera Mouse with the threshold kernel in normal sessions. Each point represents the *action* of a user in the Normal Sessions, who directs the mouse to a target, with 400 actions total. The Index of Difficulty (ID) of each action, corresponding to the size and the distance of the target. For each action, a higher ID is correlated to more time to reach the target. The Index of Performance represents the bandwidth of the device, and is the reciprocal of the slope of the best linear fit of the actions.

increased familiarity of the users with the Camera Mouse may cause them to naturally move the mouse quicker in the sessions at the end of their time with the trackers. This results in a potential source of bias for Table 6.3. To address this issue, we examined the average acceleration of mouse pointer movements. This measure is the increase in speed of the movement of the pointer controlled by subjects within a particular session and it indicates the rate of learning of the users. The average acceleration can be approximated by the slope of the best linear fit of actions in a session. Each action

Table 6.3: Index of Performance. The rows represent the type of system used for tracking. The columns represent the tracking session performed. The values represent the Index of Performance or the bandwidth of the device, in units of bits per second. The Index of Performance is roughly the ratio of the difficulty of reaching a target to the time to reach it, with larger numbers signifying better performance.

Index of Performance (Bits Per Second)					
Tracker	Sessions				
	Normal	Hastened	Boundary	Changed Lighting	Changed Scale
Standard Camera Mouse	1.4	0.94	2.1	×	×
Threshold Kernel	2.9	0.87	2.4	×	×
Normalized Threshold Kernel	1.9	×	×	1.7	×
Normal. Radial Intensity Kernel	1.7	×	×	×	1.6

is plotted by the mouse speed of the pointer during the action (in units of pixels per second) versus the occurrence time of the action in the session (in units of seconds). The average increase of speed across all users is in units of pixels per second squared.

The average acceleration of the users was heavily correlated to the session type and not the tracker used (Table 6.4). Users had average accelerations close to zero when instructed to move naturally (in the normal and changed lighting and scale sessions), so the bias can be discounted for those sessions. Users had the same average acceleration for the boundary sessions with both systems, indicating no relative bias. Users had high average accelerations for the hastened sessions with respective rates of 6.9 and 11 pixels/s^2 for the Standard and Augmented Camera Mouse, indicating the possibility of a comparative bias between the hastened sessions. From results of Table 6.4, we showed that learning was not a significant factor for bias in Table 6.3.

6.9 Experiment with Subject with Severe Motion Impairments

We worked with a quadriplegic subject whose voluntary motion is severely limited due to a massive stroke, which had occurred four years earlier. The subject communicates with friends and family members through eye and eyebrow motions. In our

Table 6.4: Learning Bias in Index of Performance Measurements. The rows represent the type of Kernel-Subset-Kernel used for tracking. The columns represent the tracking session performed. The values represent the average acceleration of mouse pointer speed of the subjects, which is used to approximate the average rate of learning of the subjects. The results show that the learning of the subjects had minimal impact on the bias of the Indices of Performance measurements (Table 6.3).

Average Acceleration of Mouse Pointer (Pixels / s^2)					
Tracker	**Sessions**				
	Normal	Hastened	Boundary	Changed Lighting	Changed Scale
Standard Camera Mouse	-0.06	6.9	5.4	×	×
Threshold Kernel	-0.23	11	5.4	×	×
Normalized Threshold Kernel	0.14	×	×	0.64	×
Normal. Radial Intensity Kernel	0.82	×	×	×	0.35

experiments, we used a blink detection method [Epstein and Betke (2012)] to automatically find the eyes of the subject and then tracked the subject's eyebrow motion with the Augmented Camera Mouse. Since the eyebrow motion was mostly vertical, see Figure 6·10, the conversion of this motion into mouse pointer coordinates would only enable up- and down cursor movements. We needed to adjust our experiment to the subject's movement abilities. We therefore simplified the interaction mechanism and worked with test programs that only required mouse clicks and not mouse pointer positions as inputs. Our system automatically interpreted raised eyebrows as mouse clicks. Click events were sent to a text-entry program called Customizable Keyboard [Epstein and Betke (2012)]. Customizable Keyboard is a scan-based on-screen keyboard that can be adapted to the user's motion abilities. It is similar to virtual scanning keyboards analyzed by [Bhattacharya et al. (2008)]. Using the Augmented Camera Mouse with the Customizable Keyboard, the subject was able to spell out words by raising his eyebrows and thereby selecting highlighted letters during a scan of the alphabet.

The eyebrow was tracked using the Augmented Camera Mouse, in the same configuration as described in Section 6.8.4. The Kernel-Subset-Tracker was used with the threshold kernel. A training set of size 25 was used with a real-time subset of

Figure 6·10: The Augmented Camera Mouse was used to track the eyebrow of a subject with movement disabilities. The vertical motions of the subject's eyebrow were translated into mouse click events.

size 5. The training set consisted of images of size 100×100 centered at the subject's eyebrow.

The user task during the training phase, as described in Section 6.7, had to be adjusted for our subject due to his limited movement abilities. To enable the Augmented Camera Mouse to collect training images, we asked the user to look at the camera, blink a few times, and then raise his eyebrows. The central location of the eyebrow was detected using an automatic feature locator that is based on a blink detection method [Epstein and Betke (2012)] A representative set of images of the subject's eyebrow in the raised and normal states was collected every second for 25 seconds while the subject moved his eyebrows up and down.

During the test phase of the experiment, the subject generated click events by raising and lowering his eyebrows. Upward motions of the tracked feature on the eyebrow would trigger a click event (Figure 6·11). In every frame, the system determines the vertical difference Y between the position of the eyebrow in current frame and in the previous frame. The "raw Y movement" is smoothed using a moving average of period 20 with exponentially decreasing weights.

Before the subject could use the Augmented Camera Mouse as an interface, we needed to specify a threshold for the range of motion that was comfortable for the him and that could be mapped accurately to a click command. We set the click threshold

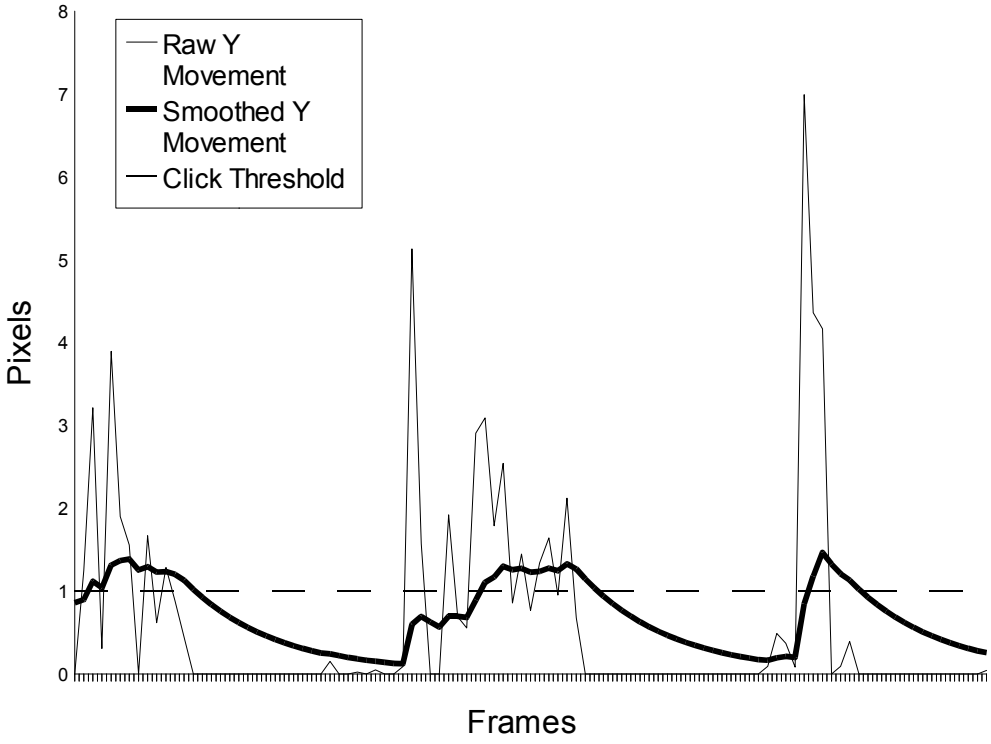

Figure 6·11: The difference in Y positions of the feature between frames is represented by "Raw Y Movement". This value is smoothed using an exponential average, as represented by the "Smooth Y Movement". Click events are generated when the smoothed y movement first transitions from under the click threshold to over it. In the example above, three clicks were generated.

manually using the pop-up window shown in Figure 6·12.

The subject used the Augmented Camera Mouse in two test sessions. The first session lasted 4.7 minutes and the second session lasted 6.9 minutes. The Augmented Camera Mouse successfully tracked the user's eyebrow. The user was able to communicate by raising his eyebrow and selecting letters, spelling out words, and creating sentences.

To evaluate the tracking accuracy of the Augmented Camera Mouse, we compared computed feature positions against manual "ground-truth" markings of feature locations. For each session, an image from the webcam was saved once per second. After the session was over, an independent observer used a custom program to mark the

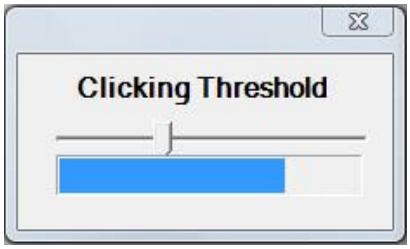

Figure 6·12: A pop-up window is used to determine the user-specific threshold for click events. The blue bar represents the vertical distance of the tracked eyebrow from its neutral position. The slider knob pointing down represents the position of the click threshold. When the blue motion bar transitions from the left to the right of this position, a mouse click is generated. At the beginning of the experiment, we asked the subject to make eyebrow movements as if he intended to generate mouse clicks. By observing the range of the blue bar during his movements, we could determine a click-threshold position, as shown here, that was comfortable and effective for him.

location of the facial feature in each image. For both sessions, the average Euclidean distance between the target locations and the manually marked locations was computed. We also computed the feature drift, as defined in Section 6.8.4.

Our results (Table 6.5) show that the subject's eyebrow was tracked accurately by the Augmented Camera Mouse for the duration of the two test sessions. The average pixel error was very small and the feature drift was minimal.

Table 6.5: Results of eyebrow clicking experiment with user with severe motion disabilities.

	Eyebrow Tracking Sessions	
	Session 1	Session 2
Duration	4.7 min	6.9 min
Number of manually marked images	270	403
Average tracker error	4.6 pixels	4.2 pixels
Feature drift	0.0 pixels per second	0.0 pixels per second

6.10 Conclusions

We introduced the KERNEL-SUBSET-TRACKER, an exemplar tracker that uses kernel methods traditionally associated with classification. We showed that the KERNEL-SUBSET-TRACKER can maintain a sufficiently reliable tracking performance with a subset size of 5, given 25 training observations. The setup phase of the KERNEL-SUBSET-TRACKER is efficient and can be accomplished in real time.

We showed how the standard threshold kernel can be "normalized" to provide invariance to linear changes in brightness and contrast. As shown experimentally, the Normalized Radial Intensity Kernel is invariant to changes in scale. The NRI Kernel is computationally more expensive than the other two kernels, but it still maintains the same frame rate as the other kernels when used by the Augmented Camera Mouse. The use of the NRI Kernel is recommended in interaction scenarios where the user may move significantly towards or away from the camera. Additional kernels may be developed in the future that enable to the KERNEL-SUBSET-TRACKER to achieve invariance to other object transformations that represent user movement.

Our experimental results show that the Augmented Camera Mouse had no significant feature drift, and therefore was anchored to a particular feature, regardless of fast movement or extreme head positions. This is an improvement to the Standard Camera Mouse, which was subject to feature drift, even in the "normal" test sessions.

We tested the Augmented Camera Mouse with a user with severe movement disabilities. The Augmented Camera Mouse was shown to track the subject's eyebrow accurately, enabling him to communicate via mouse click events.

References

ABLEDATA (2010). Online database that provides information on assistive technology, U.S. Department of Education. http://www.abledata.org.

Akram, W., Tiberii, L., and Betke, M. (2006). A customizable camera-based human computer interaction system allowing people with disabilities autonomous hands free navigation of multiple computing tasks. In Stephanidis, C. and Pieper, M., editors, *Universal Access in Ambient Intelligence Environments – 9th International ERCIM Workshop "User Interfaces For All" UI4ALL 2006, Königswinter, Germany, September 2006, Revised Papers. LNCS 4397*, pages 28–42. Springer-Verlag, Berlin.

Bates, R. and Istance, H. (2003). Why are eye mice unpopular? A detailed comparison of head and eye controlled assistive technology pointing devices. *Universal Access in the Information Society*, 2(3):265–279.

Belkin, M. and Niyogi, P. (2003). Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.

Bengio, Y., Paiement, J.-F., and Vincent, P. (2003). Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In *Advances in Neural Information Processing Systems*, pages 177–184.

Betke, M. (2009). Intelligent interfaces to empower people with disabilities. In Nakashima, H., Augusto, J. C., and Aghajan, H., editors, *Handbook of Ambient Intelligence and Smart Environments*. Springer Verlag, New York, Dordrecht, Heidelberg, London.

Betke, M., Gips, J., and Fleming, P. (2002). The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(1):1–10.

Betke, M., Gusyatin, O., and Urinson, M. (2006). SymbolDesign: A user-centered method to design pen-based interfaces and extend the functionality of pointer input devices. *Universal Access in the Information Society*, 4(3):223–236.

Bhattacharya, S., Samanta, D., and Basu, A. (2008). Performance models for automatic evaluation of virtual scanning keyboards. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(5):510–519.

Biswas, P. and Samanta, D. (2007). Designing Computer Interface for Physically Challenged Persons. In *Proceedings of the International Information Technology Conference (ICIT 2007)*, pages 161–166, New York, NY. IEEE.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965.

Bourel, F., Chibelushi, C., and Lowe, A. (2000). Robust Facial Feature Tracking. In *Proceedings of the British Machine Vision Conference (BMVC)*, U.K. BMVA Press. 10 pp.

Chitta, R., Jin, R., Havens, T., and Jain, A. (2011). Approximate Kernel k-means: Solution to Large Scale Kernel Clustering. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 895–903. ACM.

Cox, T. and Cox, M. (2000). *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC.

Epstein, S. and Betke, M. (2009). Active hidden models for tracking with kernel projections. Technical Report BU-CS-TR-2009–006, Department of Computer Science, Boston University.

Epstein, S. and Betke, M. (2011a). An Information Theoretic Representation of Agent Dynamics as Set Intersections. In *Proceedings of the Fourth Conference on Artificial General Intelligence*, volume 6830 of *Lecture Notes in Artificial Intelligence*, pages 72–81. Springer.

Epstein, S. and Betke, M. (2011b). The Kernel Semi-least Squares Method for Sparse Distance Approximation. Technical report, Department of Computer Science, Boston University.

Epstein, S. and Betke, M. (2012). The kernel semi-least squares method for distance approximation. Unpublished manuscript.

Epstein, S., Hescott, B., and Homer, S. (2011). Collaboration on the algorithmic weight of sets of reals. Unpublished Manuscript.

Epstein, S. and Levin, L. A. (2011). Sets Have Simple Members. *CoRR*, arXiv:abs/1107.1458.

Epstein, S., Missimer, E., and Betke, M. (2012). Using Kernels for a Video-Based Mouse-Replacement Interface. *Personal and Ubiquitous Computing Special Issue*.

Findlater, L., Jansen, A., Shinohara, K., Dixon, M., Kamb, P., Rakita, J., and Wobbrock, J. (2010). Enhanced area cursors: reducing fine pointing demands for people with motor impairments. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST'10)*, pages 153–162.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal Experimental Psycholology: General*, 47(6):381–391.

Gács, P. (1975). On the Symmetry of Information. *Soviet Mathematics Doklady*, 15(6):1477–1480.

Gács, P., Tromp, J., and Vitányi, P. (2001). Algorithmic Statistics. *IEEE Transactions on Information Theory*, 47(6):2443–2463.

Gips, J., Betke, M., and Fleming, P. (2000). The Camera Mouse: Preliminary Investigation of Automated Visual Tracking for Computer Access. In *Proceedings of the Rehabilitation Engineering and Assistive Technology Society of North America 2000 Annual Conference (RESNA 2000)*, pages 98–100, Arlington, VA. RESNA.

Gorodnichy, D., Dubrofsky, E., and Ali, M. (2007). Working with computer hands-free using Nouse Perceptual Vision Interface. In *Proceedings of the International CRV Workshop on Video Processing and Recognition (VideoRec'07), Montreal, Candada*, Canada. NRC.

Graham, D. and Allinson, N. (1998). Characterizing Virtual Eigensignatures for General Purpose Face Recognition. In *Face Recognition: From Theory to Applications*, pages 446–456.

Hutter, M. (2004). *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin.

Hwang, F., Keates, S., Langdon, P., and Clarkson, J. (2004). Mouse movements of motion-impaired users: a submovement analysis. In *Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '04)*, pages 102–109, New York, NY. ACM.

International Organization for Standardization (2007). Ergonomics of human-system interaction - Part 400: principles and requirements for physical input devices.

iView X HED (2010). SensoMotoric Instruments. http://www.smivision.com.

Kim, H. and Ryu, D. (2006). Computer control by tracking head movements for the disabled. In *10th International Conference on Computers Helping People with Special Needs (ICCHP), Linz, Austria, LNCS 4061*, pages 709–715. Springer-Verlag, Berlin Heidelberg.

Kim, W. B., Kwan, C., Fedyuk, I., and Betke, M. (2008). Camera Canvas: Image Editor for People with Severe Disabilities. Technical Report BU-CS-TR-2008-010, Computer Science Department, Boston University.

Kjeldsen, R. (2006). Improvements in vision-based pointer control. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 189–196, New York, NY. ACM.

Levin, L. A. (1974). Laws of Information Conservation (Non-growth) and Aspects of the Foundations of Probability Theory. *Problemy Peredachi Informatsii*, 10(3):206–210.

Levin, L. A. (1984). Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37.

Levin, L. A. (2002). Forbidden information. *CoRR*, cs.CC/0203029.

Li, M. and Vitányi, P. (2008). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition.

Loewenich, F. and Maire, F. (2007). Hands-free mouse-pointer manipulation using motion-tracking and speech recognition. In *Proceedings of the 19th Australasian Conference on Computer-Human Interaction (OZCHI)*, pages 295–302, New York, NY. ACM.

Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the Seventh International Joint Conference on Artifical Intelligence*, pages 674–679, California. IJCAI.

MacKenzie, I. S. (1989). A note on the information-theoretic basis for Fitts' law. *Journal of Motor Behavior*, 21:323–330.

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human Computer Interaction*, 7(1):91–139.

Magee, J. and Betke, M. (2010). HAIL: hierarchical adaptive interface layout. In et al., K. M., editor, *12th International Conference on Computers Helping People with Special Needs (ICCHP 2010), Vienna University of Technology, Austria, Part 1, LNCS 6179*, pages 139–146. Springer-Verlag, Berlin Heidelberg.

Manresa-Yee, C., Varona, J., Perales, F., Francisco, J., Negre, F., and Muntaner, J. (2008). Experiences using a hands-free interface. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 261–262, New York, NY. ACM.

Palleja, T., Rubion, E., Teixido, M., Tresanchez, M., del Viso, A., Rebate, C., and Palacin, J. (2008). Simple and robust implementation of a relative virtual mouse controlled by head movements. In *Proceedings of the Conference on Human System Interactions*, pages 221–224, Piscataway, NJ. IEEE.

Ponsa, P., Manresa-Yee, C., Batlle, D., and Varona, J. (2009). Assessment of the use of a human-computer vision interaction framework. In *Proceedings of the 2nd Conference on Human System Interactions*, pages 431–438, Piscataway, NJ. IEEE.

QualiEye (2010). Qualilife. http://www.qualilife.com/en/accessibility-download.html.

Quick Glance 3 (2010). EyeTech Digital Systems. http://www.eyetechds.com.

Radhakrishna, R. and Mitra, S. (1971). Further Contributions to the Theory of Generalized Inverse of Matrices and Its Applications. *The Indian Journal of Statistics, Series A*, 33(3):289–300.

RED Eye Tracking System (2010). SensoMotoric Instruments. http://www.smivision.com.

Roweis, S. and Saul, L. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition.

Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K., Rätsch, G., and Smola, A. (1999). Input Space Versus Feature Space in Kernel-Based Methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017.

Schölkopf, B. and Smola, A. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press.

Shen, A. (1983). The concept of (alpha,beta)-stochasticity in the Kolmogorov sense, and its properties. *Soviet Mathematics Doklady*, 28(1):295–299.

Shen, A. (1999). Discussion on Kolmogorov Complexity and Statistical Analysis. *The Computer Journal*, 42(4):340–342.

Sim, T., Baker, S., and Bsat, M. (2002). The CMU Pose, Illumination, and Expression (PIE) Database. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*.

SmartNAV (2010). NaturalPoint. http://www.naturalpoint.com/smartnav.

Solomonoff, R. J. (1964). A Formal Theory of Inductive Inference, Part l. *Information and Control*, 7:1–22.

Talwalkar, A., Kumar, S., and Rowley, H. (2008). Large-Scale Manifold Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008*, pages 1 –8.

Tenenbaum, J., Silva, V., and Langford, J. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323.

Trewin, S., Keates, S., and Moffatt, K. (2006). Developing Steady Clicks:: A Method of Cursor Assistance for People with Motor mpairments. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '06)*, pages 26–33, New York, NY. ACM.

Tu, J., Tao, H., and Huang, T. (2007). Face as mouse through visual face tracking. *Computer Vision and Image Understanding*, 108(1-2):35–40.

Varona, J., Manresa-Yee, C., and Perales, F. (2008). Hands-free vision-based interface for computer accessibility. *Journal of Network and Computer Applications*, 31(4):357–374.

Vereshchagin, N. and Vitányi, P. (2004a). Algorithmic Rate Distortion Theory. http://arxiv.org/abs/cs.IT/0411014.

Vereshchagin, N. and Vitányi, P. (2004b). Kolmogorov's Structure Functions and Model Selection. *IEEE Transactions on Information Theory*, 50(12):3265 – 3290.

Vereshchagin, N. and Vitányi, P. (2010). Rate Distortion and Denoising of Individual Data using Kolmogorov Complexity. *IEEE Transactions on Information Theory*, 56.

V'Yugin, V. (1987). On Randomness Defect of a Finite Object Relative to Measures with Given Complexity Bounds. *SIAM Theory of Probability and Its Applications*, 32:558–563.

V'Yugin, V. (1999). Algorithmic complexity and stochastic properties of finite binary sequences.

Waber, B., Magee, J., and Betke, M. (2006). Web mediators for accessible browsing. In Stephanidis, C. and Pieper, M., editors, *Universal Access in Ambient Intelligence Environments – 9th International ERCIM Workshop "User Interfaces For All" UI4ALL 2006, Königswinter, Germany, September 2006, Revised Papers. LNCS 4397*, pages 447–466. Springer-Verlag, Berlin.

Weiss, Y. (1999). Segmentation using eigenvectors: A unifying view. *IEEE International Conference on Computer Vision*, 2:975.

Williams, C. and Seeger, M. (2001). Using the Nyström Method to Speed Up Kernel Machines. In Todd K. Leen, T. G. D. and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13, pages 682–688.

Wobbrock, J. and Gajos, K. (2008). Goal crossing with mice and trackballs for people with motor impairments: Performance, submovements, and design directions. *ACM Transactions on Accessible Computing*, 1(1):1–37.

Yang, L. and Jin, R. (2006). Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University.

CURRICULUM VITAE