DISCRIMINATIVE MODELS AND DIMENSIONALITY REDUCTION FOR REGRESSION

BY MINYOUNG KIM

A dissertation submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
Graduate Program in Computer Science

Written under the direction of
Vladimir Pavlovic

and approved by

New Brunswick, New Jersey
May, 2008
ABSTRACT OF THE DISSERTATION

Discriminative Models and Dimensionality Reduction for Regression

by Minyoung Kim

Dissertation Director: Vladimir Pavlovic

Many prediction problems that arise in computer vision and robotics can be formulated within a regression framework. Unlike traditional regression problems, vision and robotics tasks are often characterized by a varying number of output variables with complex dependency structures. The problems are further aggravated by the high dimensionality of the input. In this thesis, I address two challenging tasks related to learning of regressors in such settings: (1) developing discriminative approaches that can handle structured output variables, and (2) reducing the dimensionality of the input while preserving the statistical correlation with the output.

A complex dependency structure in the output variables can be effectively captured by probabilistic graphical models. In contrast to traditional joint data modeling for probabilistic models, I propose conditional models and a discriminative learning approach that are directly related to the ultimate prediction objective. While discriminative learning of structured models such as Conditional Random Fields (CRFs) has attracted significant interest in the past, learning structured models in the regression setting has been rarely explored. In this work I first extend the CRF and the discriminatively trained HMM methods to the structured output regression problem. I propose
two different approaches based on directed and undirected models. In the second approach the parameter learning is cast as a convex optimization problem, accompanied by a new approach that effectively handles the density integrability constraint. Experiments in several problem domains, including human motion and robot-arm state estimation, indicate that the new models yield high prediction accuracy comparable to or better than state-of-the-art approaches.

In the second part of the thesis, I consider the task of finding a low-dimensional representation of the input covariates while preserving the statistical correlation in regressing the output. This task, known as the dimensionality reduction for regression (DRR), is particularly useful when visualizing high-dimensional data, efficiently designing regressors with a reduced input dimension, and eliminating noise in the input data by uncovering essential information for predicting the output. While the dimensionality reduction methods are common in many machine learning tasks, their use in the regression settings has not been widespread. A number of recent methods for DRR have been proposed in the statistics community but suffer from several limitations, including non-convexity and the need for slicing of potentially high-dimensional output space. I address these issues by proposing a novel approach based on covariance operators in reproducing kernel Hilbert spaces (RKHSes) that provide a closed-form DRR solution without the need for explicit slicing. The benefits of this approach are demonstrated in a comprehensive set of evaluations on several important regression problems in computer vision and pattern recognition.
Acknowledgements

The journey of writing a dissertation is a challenging experience, but it would not have been possible without the support and help of numerous people. Thus, my sincere gratitude goes to everyone whom I would like to mention.

I would like to sincerely thank my advisor Prof. Vladimir Pavlovic, for his guidance, encouragement, patience, and understanding during my graduate studies at Rutgers University. He is a great mentor and brought me into the pleasure of research. He has provided me with the opportunity to individuality and self-sufficiency by allowing me to work with such independence.

I thank the thesis committee, Prof. Dimitris Metaxas, Prof. Ahmed Elgammal, and Dr. Sanjiv Kumar, for their insightful and detailed comments on this thesis work.

During the summer internship at Google Research, I am indebted to many excellent researchers for their priceless advice: Corina Cortes, Sergey Ioffe, Sanjiv Kumar, Henry Rowley, and Ming Zhao.

I thank all faculty members and graduate students in Rutgers computer science department.

I would like to thank my professors in Korea for helping me receive my doctoral degree: Prof. Yookun Cho, Prof. Sukho Lee, and Prof. Hyunsik Shin.

Finally, and most importantly, I am especially grateful to my family. Minjung’s warm support and encouragement made my long journey happy and amusing. I thank my parents for their ceaseless support. Thanks to my brother, sister-in-law, and lovely niece, Hyunju.
Dedication

To my family.
Table of Contents

Abstract ................................................................. ii
Acknowledgements ......................................................... iv
Dedication ................................................................. v
List of Tables .............................................................. ix
List of Figures .............................................................. x

1. Introduction ............................................................. 1
   1.1. Traditional Approaches to Regression ......................... 2
   1.2. Structured Multiple Output Regression ......................... 3
      1.2.1. Probabilistic Model Based Approach ....................... 4
   1.3. Dimensionality Reduction for Regression ....................... 6

2. Generative Models vs. Discriminative Models ....................... 8
   2.1. Example Models for Classification ............................. 9
      2.1.1. Naïve Bayes vs. Logistic Regression .................... 9
      2.1.2. Hidden Markov Models vs. Conditional Random Fields .... 11
   2.2. Why Discriminative Models? .................................. 16
      2.2.1. Example: Generative vs. Conditional Gaussian Mixture .... 18
   2.3. Discriminative Learning of Generative Models ................. 23
      2.3.1. Conditional Likelihood Maximization .................... 25
      2.3.2. Example: Classification with Mixtures of Gaussians .... 27
      2.3.3. Evaluation on Real Data Classification .................. 31

3. Discriminative Learning of Dynamical Systems ..................... 33
3.1. State-Space Model ................................................. 35
3.2. Difficulty in Discriminative Dynamic Modeling ................. 37
  3.2.1. Conditional Random Fields ................................ 37
  3.2.2. Maximum Entropy Markov Models .......................... 38
3.3. Discriminative Learning of State-Space Models .................. 39
  3.3.1. Conditional Likelihood Maximization (CML) ............... 40
  3.3.2. Slicewise Conditional Likelihood Maximization .......... 41
  3.3.3. Extension to Nonlinear Dynamic Models ................... 44
3.4. Related Work ...................................................... 45
3.5. Evaluation ............................................................ 45
  3.5.1. Synthetic Data ................................................. 46
  3.5.2. Human Motion Data ............................................ 46

4. Conditional State-Space Models ...................................... 51
  4.1. Conditional State-Space Model (CSSM) ......................... 52
    4.1.1. Feasible Parameter Space ................................. 54
    4.1.2. Parameter Learning ...................................... 54
    4.1.3. Inference and Decoding .................................... 58
  4.2. Variants of CSSM ............................................... 60
    4.2.1. Discriminative Feature Selection ....................... 60
    4.2.2. Nonlinear Dynamics ...................................... 61
  4.3. Related Work ..................................................... 61
  4.4. Evaluation ........................................................ 62
    4.4.1. Synthetic Data ............................................ 62
    4.4.2. Robot-Arm Data ........................................... 63
    4.4.3. Human Motion Data ....................................... 67

5. Dimensionality Reduction for Regression .......................... 68
  5.1. Background ....................................................... 71
    5.1.1. Sliced Inverse Regression (SIR) ......................... 71
5.1.2. Kernel Extension of Inverse Regression ................. 72
5.2. Proposed Approach .............................................. 74
  5.2.1. IR using Covariance Operators in RKHS .............. 74
  5.2.2. KSIR as a Special Case of COIR .................. 76
5.3. Empirical Evaluation ........................................... 78
  5.3.1. Synthetic Curves Dataset ......................... 78
  5.3.2. Head Pose Estimation ............................... 79
  5.3.3. Noisy Sparse Head Pose Estimation ............. 81
  5.3.4. Human Figure Pose Estimation .................. 84
  5.3.5. Hand-written Digit Image Reconstruction ....... 86
  5.3.6. Temperature Map .................................. 89

6. Conclusions ......................................................... 92
References ............................................................. 94
Vita ................................................................. 98
List of Tables

2.1. Prediction errors of generative and conditional mixtures. 23
2.2. Structured output classification test accuracies (%). 32
3.1. Test errors and log-perplexities for synthetic data. 46
3.2. Average test errors. The error types are abbreviated as 3 letters: The first indicates smoothed (S) or filtered (F), followed by 2 letters meaning that the error is measured in either the joint angle space (JA) or the 3D articulation point space (3P) (e.g., SJA = smoothed error in the joint angle space). The unit scale for the 3D point space is deemed as the height of the human model $\sim 25$. 48
4.1. Test errors and log-perplexities on synthetic data. 63
4.2. Test L2 errors for robot-arm data. 65
4.3. Average test L2 errors for CMU walking motion data. 67
5.1. Test (RMS) errors in 3D human figure pose estimation. 86
5.2. Test (RMS) errors in scratched digit image denoising. 88
5.3. Temperature map: RMS errors (in K) by linear regression. 89
List of Figures

1.1. Examples of structured multiple output regression. .......................... 4
2.1. Graphical representation: Naive Bayes and Logistic Regression. ....... 9
2.2. Test errors vs. sample sizes (m) for Naive Bayes (solid lines) and Logistic Regression (dashed lines) on UCI datasets. Excerpted from [37]. .... 11
2.3. Graphical representation of HMM and CRF. ................................. 12
2.4. Test error scatter plots on synthetic data comparing HMMs and CRFs in structured output classification. The open squares represent datasets generated from $a < 1/2$, and the solid circles for $a > 1/2$. Excerpted from [27]. .............................................................. 17
2.5. Data sampled from the piecewise linear model Eq. (2.21). In (b), most data points at the central support around (2, 2) are eliminated. ......... 19
2.6. The maximum-likelihood trained generative models on $D_1$ and $D_2$. .... 21
2.7. Prediction plots and gating functions: The dashed lines in the prediction plot correspond to the linear mean functions $y = a_m x + b_m$ of the conditional mixture. The green line indicates the final prediction which is selected from two linear mean functions by $0 - 1$ thresholding of the gating function shown below. “True” in Fig 2.7(a) shows the conditional model in Eq. (2.23). For the generative mixture models, the equivalent conditional models are plotted using the conversion equations in Eq. (2.25). 22
2.8. Asymptotic behavior of generative and discriminative learning. ........ 30
3.1. Graphical models: HMM (or SSM), CRF, and MEMM. ...................... 35
3.2. Visualization of estimated sequences for synthetic data. It shows the estimated states (for dim-1) at \( t = 136 \sim 148 \). The ground truth is depicted by solid (cyan) line, ML by dotted (blue), CML by dotted-dashed (red), and SCML by dashed (black).

3.3. Skeleton snapshots for walking (a–f), picking-up a ball (g–l), and running (m–s): The ground-truth is depicted by solid (cyan) lines, ML by dotted (blue), SCML by dashed (black), and latent variable nonlinear model (LVN) by dotted-dashed (red).

4.1. Gradient search with step-size adjustment.

4.2. Feature selection. The solid (cyan) line indicates test L2 errors of the CSSM learned with the features selected by the proposed algorithm. It starts from empty feature, adds a feature at each stage, up to full features (22-nd stage). Each number indicates the feature index, where the original features correspond to 1 and 2. The test errors of the SSM learned with full 22 features are depicted: SSM-ML by dotted (blue), SSM-CML by dotted-dashed (red), and SSM-SCML by dashed (black).

4.3. Robot-Arm data samples.

4.4. Feature selection with the RBF extended feature set. The left is the dataset (b) and the right is the dataset (c). See text for details.

5.1. Central subspaces for the curves dataset.

5.2. 2D central subspaces of face images with the output comprised of 2D pose. The points are colored by the true Left/Right pose angles in (a)/(d), Up/Down in (b)/(e), and (c)/(f) shows the face images superimposed. The test points are depicted in black circles.

5.3. 3D central subspaces of face images with the output comprised of 2D pose and the lighting. The points are colored by the true Left/Right pose angles in (a)/(d), Up/Down in (b)/(e), and the lighting direction in (c)/(f). The test points are depicted in black circles.

5.4. Central subspaces for noisy sparse head pose estimation.
5.5. Test face images on the COIR central subspace (noisy sparse head pose estimation). .................................................. 85

5.6. Selected skeleton and silhouette images for a half walking cycle: From stand-up ($t = 56$), right-leg forwarding, left-leg catching-up, and back to stand-up ($t = 120$). The skeleton images are drawn using the 3D joint angles. ................................................................. 86

5.7. Central subspaces for silhouette images from walking motion: The blue (red) points indicate train (test) data points. .................................................. 87

5.8. Denoising USPS scratched digit images. Each 5-tuple is composed of, from left to right, ($1^{st}$) the noise-free test image, ($2^{nd}$) randomly scratched image, ($3^{rd}$) denoised by NN on COIR, ($4^{th}$) NN on KSIR, and ($5^{th}$) NN on the scratched image $\mathbf{x}$ itself. .................................................. 88

5.9. Temperature map: (a) is the true $\mathbf{y}$ (temperature) map. (b)-(e) plot $\mathbf{y}$ vs. dimension-reduced $\mathbf{x}$ for the competing methods (train/test points are in blue/red). (f)-(i) show the predicted temperatures using the linear regression estimation on the central subspaces. (j)-(m) show the difference maps between the true and the predicted temperatures (the larger difference, the darker). .................................................. 90
Chapter 1

Introduction

One of the fundamental problems in machine learning is to predict some unseen quantity of interest (denoted by $y \in \mathcal{Y}$) from the observation of a phenomenon (denoted by $x \in \mathcal{X}$). In the supervised statistical learning framework, the problem is to estimate a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$ (i.e., $y = f(x)$) from a finite number of complete data pairs, often called the train data, $D = \{(x^i, y^i)\}_{i=1}^n$ which are sampled independently and identically from the underlying, but unknown, distribution $P(x, y)$ on the joint probability measure space $(\mathcal{X}, \mathcal{Y})$.

In this thesis, I consider the regression\footnote{In the regression terminology, $x$ is often called \textit{input} or \textit{covariates}, while $y$ is referred to as \textit{output} or \textit{responses}.} problem where the random variables comprising $y$ take continuous values in the real domain. Many prediction problems that arise in computer vision and robotics can be formulated in a regression framework. This thesis particularly tackles two challenging tasks as follows.

1. **Structured Multiple Output Regression**: How to deal with the output $y$ composed of a (possibly varying) number of random variables related by an unknown dependency structure.

2. **Dimensionality Reduction for Regression**: How to reduce the dimensionality of the high-dimensional input $x$ while preserving its statistical correlation with the output $y$.

Despite their importance, the two problems have received less attention in the machine learning and statistics communities. Before investigating them further in Sec. 1.2 and Sec. 1.3, I briefly review the traditional approaches to regression estimation and the underlying mathematical framework.
1.1 Traditional Approaches to Regression

The regression problem can be seen as the risk minimization problem [63], which is composed of two steps: (1) Define a family of prediction functions $\mathcal{F} = \{ f : \mathcal{X} \to \mathcal{Y} \}$ either parametrically or non-parametrically, and (2) Find an optimal predictor $f^*$ in the hypothesis space $\mathcal{F}$ by solving the following optimization problem:

$$f^* = \arg \min_{f \in \mathcal{F}} \int_{\mathcal{X} \times \mathcal{Y}} L(y, f(x)) \, dP(x, y),$$

(1.1)

where $L(y, f(x))$ is a real-valued domain-specific loss function that typically assigns a higher value to the decision $f(x)$ that is less compatible with the corresponding target $y$ (e.g., the $L2$ loss $\|y - f(x)\|^2$). Since we do not know the true distribution $P(x, y)$ in general, our estimation is based on the train data $D = \{(x^i, y^i)\}_{i=1}^n$. Using the plug-in approximation of $P(x, y)$, the empirical version of Eq. (1.1) is:

$$f_{\text{emp}}^* = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(y^i, f(x^i)) + \lambda R(f).$$

(1.2)

The last term (with a balancing factor $\lambda \geq 0$), called the regularization term, is crucial for good generalization to unseen data by penalizing non-smooth functions (that have large $R(f)$) which tend to overfit to the train data $D$.

In the standard regression settings, the output $y$ is often assumed to be a single random variable (i.e., $y \in \mathcal{Y} = \mathbb{R}$). This makes it easy to define a family of predictor functions explicitly using well-known families of real-valued functions (e.g., linear, polynomial, spline, and kernel functions). One can instantiate many popular regression methods by specifying the predictor family $\mathcal{F}$, the loss function $L(y, f(x))$, and the regularizer $R(f)$. Some common cases include:

- **Ridge/LASSO Regression ([17, 58])**: Assuming that $x \in \mathbb{R}^p$, a linear function family is defined as $\mathcal{F} = \{ w = [w_1, \ldots, w_p]^T \in \mathbb{R}^p : f(x; w) = w^T \cdot x \}$. The loss function $L(y, f(x)) = (y - f(x))^2$. The ridge regression takes the $L2$ norm $R(w) = \|w\|_2^2 = \sum_{j=1}^p w_j^2$, while the LASSO takes the $L1$ norm $\|w\|_1 = \sum_{j=1}^p |w_j|$.

- **Support Vector Regression (SVR) ([63])**: With the linear function family and the $L2$ regularization, it takes the so-called $\epsilon$-insensitive (or $\epsilon$-tube) loss
\[ L(y, f(x)) = V_{\epsilon}(y - f(x)) \] where \( V_{\epsilon}(r) = |r| - \epsilon \) if \( |r| \geq \epsilon \), and \( V_{\epsilon}(r) = 0 \) otherwise. This effectively ignores the error of size smaller than \( \epsilon \), which can be seen as a natural extension of the margin-based support vector machine (SVM) in classification.

- **Piecewise Polynomials or Splines ([7])**: The prediction function family can be enriched by admitting nonlinear functions. A straightforward way is to represent a function as a linear combination of nonlinear basis functions, namely

\[
f(x) = \sum_{m=1}^{M} \beta_m h_m(x),
\]

where the basis function \( h_m(x) \) has a form of a piecewise polynomial or a spline.

- **Kernel Regressor ([65])**: A more general approach to nonlinear prediction functions is to use the kernel trick. The original input \( x \) is mapped to \( \Phi(x) \in H_k \), where \( H_k \) is a (possibly infinite-dimensional) Hilbert space endowed with a positive definite kernel \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \). Since the dot-product in the mapped space (also called the feature space) corresponds to the kernel evaluation in the original space, one can achieve a nonlinear formulation (in the original space) without modifying a linear algorithm except for the dot-product replaced by the kernel. The kernel trick can be alternatively written in a basis form, namely

\[
f(x) = \sum_{i=1}^{n} \alpha_i k(x^i, x),
\]

where the norm of \( f(\cdot) \in H_k \) is now \( R(f) = \|f(\cdot)\|^2 = \sum_{i,j} \alpha_i \alpha_j k(x^i, x^j) \).

### 1.2 Structured Multiple Output Regression

Consider a regression problem where the output \( y \) is composed of a (possibly varying) number of random variables among which some underlying dependency structure exists. A typical example is the *dynamical state regression* in time-series data: Given a measurement sequence \( x = x_1, ..., x_T \ (x_t \in \mathbb{R}^p) \), predict a corresponding sequence of real-valued multivariate states \( y = y_1, ..., y_T \ (y_t \in \mathbb{R}^d) \), where the states are correlated by some unknown dynamics.

Two interesting applications that give rise to this problem in computer vision and robotics are the *Human Motion Estimation* and the *Robot State Estimation* (Fig. 1.1).
In the human motion (e.g., walking) estimation, $x_t$ is the silhouette image at time $t$, and $y_t$ is the body pose at $t$ often represented by a set of 3D joint angles at the articulation points of a human body. (Note that $y_t$ is visualized as a skeleton image in Fig. 1.1(a).)

In the robot state estimation, we predict the dynamically changing state of a robot (e.g., position, velocity, and gear rotation angles) from the sensor observation. In the simple robot-arm case (Fig. 1.1(b)), the state $y_t$ are the arm angles at the joints, and the observation $x_t$ is the displacement of the arm end effector.

Although these problems can be tackled by traditional regression approaches within a static framework (regressing from $x_t$ to $y_t$, independently for $t$), ignoring the important information about the output dynamics would yield a suboptimal solution. Moreover, the slicewise mapping from $x_t$ to $y_t$ is often one-to-many, which increases the uncertainty in prediction (e.g., the silhouette image in the human motion is invariant to left/right arm/leg pose switching, and in the robot-arm case multiple joint angle configurations may correspond to the same displacement).

### 1.2.1 Probabilistic Model Based Approach

In order to capture the underlying dependency structure in the output, it is important to treat the output sequence as a whole. An effective way to achieve this is to have
a probabilistic model. Apart from directly estimating the predictor function, in the probabilistic approach we model the data distribution, either jointly $P(\mathbf{x}, \mathbf{y})$ or conditionally $P(\mathbf{y}|\mathbf{x})$. Accordingly, the regression estimation is turned into a problem of joint or conditional density estimation from data. Once the model is learned, the target prediction for a new input $\mathbf{x}$ is made by the so-called Maximum-A-Posteriori (MAP) decision rule, namely, $\mathbf{y} = \arg \max_{\mathbf{y}' \in \mathcal{Y}} P(\mathbf{y}'|\mathbf{x})$, where $P(\mathbf{y}'|\mathbf{x})$ can be either obtained from probabilistic inference on the joint model $P(\mathbf{x}, \mathbf{y})$, or directly available from the conditional model $P(\mathbf{y}|\mathbf{x})$.

The study of probabilistic models has long history in modern statistics. It is also known as the graphical model in the machine learning community, where rich theories and algorithms for inference and learning have been developed. Refer to the excellent tutorial [20]. In the graphical model, learning denotes determining the structure and the parameters of the model. A graph structure defines a family of probability distributions that satisfy certain conditional independencies encoded by the graph, while the parameters assigned on the graph specify a particular distribution in the family.

In the graphical representation, the nodes correspond to random variables, and the conditional independencies among the random variables (nodes) are specified by the absence of edges. The edges could be either directed (Bayesian Networks) or undirected (Markov Networks). Although the interpretation of conditional independencies in the two networks are slightly different, they can be commonly interpreted as: for three disjoint sets of random variables $A$, $B$, and $C$, the conditional independence between $A$ and $B$ given $C$ is implied by non-existence of paths between $A$ and $B$ that do not pass through any node in $C$. For instance, the chain structure in Fig. 1.1 defines a family of distributions that conform to the 1st-order Markovian dynamics (i.e., $\mathbf{y}_t \perp \mathbf{y}_{t-j} | \mathbf{y}_{t-1}$ for all $j > 1$).

The graph structure is either determined using domain knowledge, or learned from data. In the latter case, one often resorts to exhaustive or greedy incremental search [2, 43, 49]. While the structure learning is by itself an important topic, in this thesis I assume that the model structure is known (possibly incorrectly), such as the 1st-order Markovian dynamics on the output variables for the dynamical state regression
problem.

Given the structure, the parameter learning takes place. The issue of parameter-
ization is tightly coupled with the choice of the model: either a joint model \( P(x, y) \)
or a conditional model \( P(y|x) \). The joint models are often called generative models,
while the conditional models are called discriminative models. The generative model
jointly represents both input and output, while the discriminative model solely focuses
on the functional relationship between input and output. In the classification settings
(i.e., \( y \) is discrete), there has been a considerable amount of research work showing
that in many situations the discriminative models outperform the generative models in
prediction tasks [26, 27, 37].

In Chap. 2, I will review some relevant aspects of generative/discriminative mod-
els, and discuss about the discriminative learning of generative models, an alternative
way to achieve discriminative models from generative model setting. In Chap. 3 and
Chap. 4, I tackle the dynamical state regression problems using discriminative models
by addressing several known technical difficulties such as the density integrability issue.

1.3 Dimensionality Reduction for Regression

Another important and challenging issue in regression is the ability to handle high
dimensional input. The task of dimensionality reduction for regression (DRR) is to find
a low-dimensional representation \( z \in \mathbb{R}^q \) of the input covariates \( x \in \mathbb{R}^p \), with \( q \ll p \),
for regressing the output \( y \in \mathbb{R}^d \). DRR is mainly useful for: (1) visualization of high-
dimensional data, (2) efficient regressor design with a reduced input dimension, and (3)
elimination of noise in data \( x \) by uncovering the essential information \( z \) for predicting \( y \).

While dimensionality reduction methods are common in many machine learning tasks
such as discriminant analysis, graph embedding, metric learning, principal subspace
methods, their use in regression settings has not been widespread.

The crucial notion related to DRR is the sufficiency in dimension reduction (SDR, [5,
12, 33]), which states that one has to find the linear subspace bases \( B = [b_1, \ldots, b_q] \)
with \( b_l \in \mathbb{R}^p \), (in the nonlinear case, \( B = \{b_1(\cdot), \ldots, b_q(\cdot)\} \), where \( b_l(\cdot) \) is a nonlinear
basis function) such that \( y \perp x \mid B^\top x \). As this condition implies that the conditional distribution of \( y \) given \( x \) equals to that of \( y \) given \( z = B^\top x \), the dimension reduction entails no loss of information for the purpose of regression. The minimal subspace with this property is called the *central subspace*.

A number of methods originating in the statistics community have tackled the task of recovering the central space. The kernel dimension reduction (KDR) [12] and the manifold KDR [38] directly reduce the task of imposing conditional independence to the optimization problem that minimizes the conditional covariance operator in RKHS (reproducing kernel Hilbert space). However, the manifold KDR introduces a tight coupling between the central space and the separately learned input manifold, which restricts its applicability to transductive settings. Moreover, both methods introduce non-convex objectives, potentially suffering from existence of local minima.

An alternative inverse regression (IR) approach [33, 70] exploits the fact that the inverse regression \( \mathbb{E}[x|y] \) can lie on the subspace spanned by \( B \), leading to the possibility of estimating \( B \) from the slice-driven covariance estimates of the IR. While KSIR [70] overcomes the linearity of SIR [33] its performance may still suffer from the need for target \( y \) slicing, which can be unreliable for high-dimensional targets.

In this thesis I propose the Covariance Operator based Inverse Regression (COIR), a novel nonlinear method for DRR that jointly exploits the kernel Gram matrices of both input and output. COIR estimates the variance of the inverse regression under the IR framework and, at the same time, avoids the slicing by the effective use of covariance operators in RKHS. While this approach generalizes that of KSIR (a special case of COIR), it also allows a closed-form solution to the nonlinear central subspace estimation problem. Detailed study of DRR is presented in Chap. 5.

---

\(^2\) Although the term *subspace* is typically meant for a linear case, however, I abuse it referring to both linear and nonlinear cases.
Chapter 2

Generative Models vs. Discriminative Models

Discriminative models are more economic than the generative model in the sense that the former focuses only on the conditional distribution $P(y|x)$, the quantity ultimately necessary for prediction, while the latter does an unnecessary modeling effort by first estimating the full joint distribution $P(x,y)$, then inferring $P(y|x)$. However, the crucial benefit of the generative model is the ability to synthesize samples. Recently, [29] has proposed a hybrid perspective that relates generative and discriminative models by specific choices for the prior over parameters.

Although both generative and discriminative models can take a form of either Bayesian Network or Markov network, in the conventional modeling practice (assumed throughout the thesis), the generative models are represented by Bayesian Networks, while the discriminative models have undirected graphs with known $x$. Formally, $V$ be the set of nodes (random variables) in the graph, and if there exist no hidden variables, the directed generative model represents:

$$P(x,y) = \prod_{v \in V} P(v|\pi(v)),$$

(2.1)

where $v$ is a node and $\pi(v)$ indicates the parent nodes of $v$. On the other hand, the undirected discriminative model can be written as:

$$P(y|x) = \frac{\prod_C \psi_C(v_C)}{\int_y \prod_C \psi_C(v_C)},$$

(2.2)

where $C$ is a clique (complete subgraph) of the graph, $v_C$ indicates the nodes in $C$, and $\psi_C(v_C)$ is a (positive) potential function defined on clique $C$. The integral in the denominator is replaced by the sum when $y$ is discrete.

For the classification setting (i.e., discrete $y$), there has been a considerable amount of work comparing two models empirically [26, 27, 37]. In the following section, I
2.1 Example Models for Classification

2.1.1 Naive Bayes vs. Logistic Regression

In static classification (i.e., a discrete single variable $y$), the Naive Bayes (NB) and the logistic regression are two widely used instances of generative and discriminative models. As shown in Fig. 2.1, the structure of the NB indicates that the $p$ attributes of $x = [x_1, \ldots, x_p]^\top$ are independent given the class $y$. The logistic regression has a similar structure, yet it models only the conditional distribution of $y$ given $x$, assuming the attributes are always given (shaded nodes).

Despite the simple model structures, both models are known to exhibit good predictive performance in practice. For these simple generative and discriminative models, I show how they can be learned conventionally (i.e., maximum likelihood learning), and compare their generalization performance on the UCI datasets [16]. For simplicity, I assume binary classification ($y \in \{0,1\}$) and binary attributes ($x_j \in \{0,1\}$ for $j = 1,\ldots,p$).

The NB is specified by defining the local conditional distributions, $\pi = P(y = 1)$, $\mu_{j,1} = P(x_j = 1|y = 1)$, and $\mu_{j,0} = P(x_j = 1|y = 0)$, for $j = 1,\ldots,p$. The parameters of the NB are then $\theta = \{\pi, \{\mu_{j,y}\}_{j,y}\}$. Note that the parameters are probabilities thus
For i.i.d. training data \( D = \{(x^i, y^i)\}_{i=1}^n \), the log-likelihood \( LL(\theta; D) \) can be written as:

\[
LL(\theta; D) = \sum_{i=1}^n \log P(x^i, y^i | \theta) = \sum_{i=1}^n \left[ y^i \cdot \log(\pi) + (1 - y^i) \cdot \log(1 - \pi) + \sum_{j=1}^p \left( x^i_j \cdot \log(\mu_{j, y^i}) + (1 - x^i_j) \cdot \log(1 - \mu_{j, y^i}) \right) \right].
\] (2.4)

It is easy to see that \( LL(\theta; D) \) is a concave (negative convex) function of \( \theta \), and the maximum solution can be obtained analytically. To find the maximum likelihood estimator, one can differentiate \( LL(\theta; D) \) with respect to \( \theta \) and set it to zero, that is,

\[
\frac{\partial LL}{\partial \pi} = \frac{\sum_i y^i}{n} - \frac{\sum_i (1 - y^i)}{(1 - \pi)} = 0, \quad \pi_* = \frac{\sum_i y^i}{n},
\]

\[
\frac{\partial LL}{\partial \mu_{j, 1}} = \frac{\sum_{i: y^i=1} x^i_j}{\mu_{j, 1}} - \frac{\sum_{i: y^i=1} (1 - x^i_j)}{(1 - \mu_{j, 1})} = 0, \quad \mu_{j, 1}^* = \frac{\sum_{i: y^i=1} x^i_j}{\sum_{i: y^i=1} 1},
\]

\[
\frac{\partial LL}{\partial \mu_{j, 0}} = \frac{\sum_{i: y^i=0} x^i_j}{\mu_{j, 0}} - \frac{\sum_{i: y^i=0} (1 - x^i_j)}{(1 - \mu_{j, 0})} = 0, \quad \mu_{j, 0}^* = \frac{\sum_{i: y^i=0} x^i_j}{\sum_{i: y^i=0} 1}. \quad (2.5)
\]

Once the NB parameters are learned, the class prediction for a new observation \( x \) follows the MAP decision rule, \( y^* = \arg\max_{y \in \{0, 1\}} P(y | x, \theta) \).

On the other hand, the logistic regression models the conditional distribution,

\[
P(y | x, \{ w, b \}) = \frac{1}{1 + \exp ((-1)^y \cdot (w^T \cdot x + b))}.
\] (2.6)

The parameters of the logistic regression are \( \{ w, b \} \), where \( w = [w_1, \ldots, w_p]^T \in \mathbb{R}^p \) and \( b \in \mathbb{R} \). The standard learning maximizes the log-likelihood (in this case the conditional log-likelihood) \( LL(\{ w, b \}; D) \), which is:

\[
LL(\{ w, b \}; D) = -\sum_{i=1}^n \log \left[ 1 + \exp ((-1)^{y^i} \cdot (\sum_{j=1}^p w_j \cdot x^i_j + b)) \right]. \quad (2.7)
\]

While the log-likelihood function is strictly concave (having a unique maximum) in \( w \) and \( b \), however, no analytical solution exists. Instead, the gradient-based methods
In terms of classification performance, it is widely conjectured that the discriminative models outperform the generative ones even though both models have the same representational power in prediction functions. It is not difficult to show that the NB and the logistic regression share the same conditional distribution, more precisely, the logistic regression is just a reparameterization of the NB for $P(y|x)$\(^1\).

A study of analytical and empirical comparison of the NB and the logistic regression was recently conducted by [37]. They conclude that in most cases (1) the NB has the lower test error for smaller number of samples, and (2) as the sample size increases, the logistic regression eventually overtakes the NB. In other words, the logistic regression usually has lower asymptotic error than the NB, while the NB reaches its asymptotic error much faster than logistic regression. Fig. 2.2 illustrates these results on some UCI datasets.

### 2.1.2 Hidden Markov Models vs. Conditional Random Fields

One of the most popular generative sequence models is the Hidden Markov Model (HMM). In HMMs, the observation sequence $x = x_1, \ldots, x_T$ is modeled in a way that (1) a sequence of hidden states $s = s_1, \ldots, s_T$ is introduced conforming to the 1st-order

\(^1\)See Eq. (2.29) for details.
Markov transition between adjacent states $s_t$ and $s_{t+1}$, then each hidden state $s_t$ emits the observation $x_t$. The hidden states of HMMs are discrete, while the observation could be either discrete or continuous (multivariate). Hence, the transition probabilities $P(s_{t+1}|s_t)$ are multinomial distributions. The emission distributions $P(x_t|s_t)$ are multinomial for discrete $x_t$, while for continuous multivariate $x_t$, they are usually assumed to be Gaussians or the mixtures of Gaussians.

When HMMs are used for sequence-structured output classification (i.e., predicting the output sequence of discrete values, $y = y_1, \ldots, y_T$ from the observation $x = x_1, \ldots, x_T$), the hidden state sequence $s$ can be seen as the target $y$. Here I give a concrete example of HMM learning and prediction for structured output classification problems. For the sake of simplicity, I assume that we only measure $p$ binary features for the observation $x_t$. That is, we have binary predicates $f_j(x_t) \in \{0, 1\}, j = 1, \ldots, p$, as the observation at time $t$. I further assume that these $p$ binary features are independent of one another given $y_t$, namely, $P(f_1(x_t), \ldots, f_p(x_t)|y_t) = \prod_{j=1}^p P(f_j(x_t)|y_t)$. Each state $y_t$ can take a value from $\{1, \ldots, K\}$. The graphical representation of the HMM is shown in Fig. 2.3(a).

The parameters of the HMM, $\theta = \{\pi, A, \mu\}$, where $\pi_l = P(y_1 = l)$, $A_{l,l'} = P(y_t = l|y_{t-1} = l')$, and $\mu_{j,l} = P(f_j(x_t) = 1|y_t = l)$, for $l', l \in \{1, \ldots, K\}$ and $j = 1, \ldots, p$. The joint probability is:

$$P(y_1, \ldots, y_T, x_1, \ldots, x_T|\theta) = \pi_{y_1} \cdot b_{y_1}(x_1; \theta) \cdot \prod_{t=2}^T A_{y_t, y_{t-1}} \cdot b_{y_t}(x_t; \theta),$$

where $b_{y_t}(x_t; \theta)$ is the emission probability function.
where

\[ b_t(x_t; \theta) = P(f_1(x_t), \ldots, f_p(x_t)|y_t = l) = \prod_{j=1}^{p} \left[ \mu_{j,l}(x_t) \cdot (1 - \mu_{j,t})^{1-f_j(x_t)} \right], \text{ for } l = 1, \ldots, K. \]

For the train data \( D = \{(x^i, y^i)\}_{i=1}^{n} \), the joint log-likelihood \( LL(\theta; D) \) can be written as:

\[
LL(\theta; D) = \sum_{i=1}^{n} \left[ \log(\pi_{y_1^i}) + \sum_{t=2}^{T_i} \log(A_{y_t^i,y_{t-1}^i}) + \sum_{t=1}^{T_i} \log(b_{y_t^i}(x_t^i)) \right]
= \sum_{i=1}^{n} \left[ \log(\pi_{y_1^i}) + \sum_{t=2}^{T_i} \log(A_{y_t^i,y_{t-1}^i}) + \sum_{t=1}^{T_i} \sum_{j=1}^{p} \left( f_j(x_t^i) \log(\mu_{j,y_t^i}) + (1 - f_j(x_t^i)) \log(1 - \mu_{j,y_t^i}) \right) \right],
\]

where \( T_i \) denotes the length of the \( i \)-th sequence.

The maximum likelihood estimator of HMM has an analytical solution:

\[
\pi_{y_1}^* = \frac{\sum_{i=1}^{n} I(y_1^i = l)}{n}, \quad A_{y_t,y_{t-1}}^* = \frac{\sum_{i=1}^{n} \sum_{t=2}^{T_i} I(y_t^i = l, y_{t-1}^i = l')}{\sum_{i=1}^{n} \sum_{t=2}^{T_i} I(y_{t-1}^i = l')},
\]

\[
\mu_{j,l}^* = \frac{\sum_{i=1}^{n} \sum_{t=1}^{T_i} f_j(x_t^i) \cdot I(y_t^i = l)}{\sum_{i=1}^{n} \sum_{t=1}^{T_i} I(y_t^i = l)}, \text{ for } j = 1, \ldots, p, \ l, l' \in \{1, \ldots, K\}, \quad (2.9)
\]

where \( I(p) = 1 \) when the predicate \( p \) is true, and 0 when \( p \) is false.

For a new observation sequence \( x \), the output sequence prediction by the learned HMM is accomplished by the well-known HMM Viterbi decoding. It solves the optimization problem \( \arg \max_y P(y|x) \), or equivalently, \( \arg \max_y P(x, y) \) using dynamic programming: The quantity \( \delta_t(y_t) \) defined as

\[
\delta_t(y_t) = \max_{y_1,\ldots,y_{t-1}} P(y_1,\ldots,y_{t-1},y_t, x_1,\ldots,x_t), \text{ for } t = 1, \ldots, T, \quad (2.10)
\]

can be evaluated recursively, that is, (for each \( j = 1, \ldots, K \))

\[
\delta_1(y_1 = j) = P(x_1|y_1 = j) \cdot P(y_1 = j) = \pi_j \cdot b_j(x_1),
\]

\[
\delta_{t+1}(y_{t+1} = j) = \max_{y_t} \left[ P(x_{t+1}|y_{t+1} = j) \cdot P(y_{t+1} = j|y_t) \cdot \delta_t(y_t) \right] = \max_{y_t} \left[ b_j(x_{t+1}) \cdot A_{j,y_t} \cdot \delta_t(y_t) \right], \text{ for } t = 1, \ldots, T - 1. \quad (2.11)
\]
The optimal label sequence is found by backtracking: $y^*_T = \arg \max_{y_T} \delta_T(y_T)$, and $y^*_t = \arg \max_{y_t} b_{y_{t+1}}(x_{t+1}) \cdot A_{y_{t+1}, y_t} \cdot \delta_t(y_t)$ for $t = T - 1, \ldots, 1$.

Recently, discriminative models for structured output classification problems have attracted significant interests. The conditional random fields (CRFs) are the most successful conditional models, which are shown to outperform HMMs in many applications including the part-of-speech tagging, the named-entity recognition, and the protein secondary structure prediction [27, 28]. CRFs are (undirected) log-linear models that are conditioned on the observation sequence $x$ (Fig. 2.3(b)). One can form CRFs with arbitrarily complex structures by including necessary features or edges, however, at the expense of inference costs. Here, for a fair comparison, the CRF is assumed to contain only the features related with the transition and the emission of HMM’s.

The features of CRF at time $t$ are the (1st-order) transition $I(y_t = l, y_{t-1} = l')$ and the association $I(y_t = l, f_j(x_t) = 1)$, for $j = 1, \ldots, p$, $l, l' \in \{1, \ldots, K\}$, where $I(\cdot)$ is a predicate indicator as defined previously. This creates a CRF functionally equivalent to the HMM\(^2\). As a log-linear model, one associates a linear coefficient (a parameter) for each feature. Assuming homogeneous parameters, let $\lambda_{j,l}$ be the parameter for the emission feature $I(y_t = l, f_j(x_t) = 1)$, and $\eta_{l,l'}$ for the transition feature $I(y_t = l, y_{t-1} = l')$, for all $t$. The clique potential function at time $t$ can be defined as:

$$M_t(y_t, y_{t-1}|x) = \exp \left( \sum_{j=1}^{p} \sum_{l=1}^{K} \lambda_{j,l} \cdot I(y_t = l, f_j(x_t) = 1) + \sum_{l=1}^{K} \sum_{l'=1}^{K} \eta_{l,l'} \cdot I(y_t = l, y_{t-1} = l') \right).$$

(2.12)

The conditional model is represented as a product of these clique potential functions via normalization to make it a distribution for $y$, that is,

$$P(y|x) = \frac{1}{Z(x)} \prod_t M_t(y_t, y_{t-1}|x), \quad \text{where} \quad Z(x) = \sum_y \prod_t M_t(y_t, y_{t-1}|x).$$

(2.13)

At first glance, what is called the \textit{partition function}, $Z(x)$, looks infeasible to compute since the sum is over all label sequences which are exponentially many (i.e., $K^T$). However, the linear chain structure enables efficient forward/backward recursion. The

\(^2\)In essence, the CRF with the HMM features is just a reparameterization of the HMM.
forward message $\alpha_t(y_t|x)$ at time $t$ is defined as:

$$
\alpha_t(y_t|x) = \sum_{y_{t-1}} \prod_{t'=1}^{t} M_{t'}(y_{t'}, y_{t'-1}).
$$

(2.14)

From the definition, it is easy to derive the following recursion formula:

$$
\alpha_t(y_t|x) = \sum_{y_{t-1}} \alpha_{t-1}(y_{t-1}) \cdot M_t(y_t, y_{t-1}|x).
$$

(2.15)

Once the forward messages are evaluated for all $t$, the partition function can be obtained from $Z(x) = \sum_{y_T} \alpha_T(y_T|x)$. For the inference in CRF, one further needs to define the backward message $\beta_t(y_t|x)$ as:

$$
\beta_t(y_t|x) = \sum_{y_{t+1}} \cdots \sum_{y_T} \prod_{t'=t+1}^{T} M_{t'}(y_{t'}, y_{t'-1}|x).
$$

(2.16)

The backward recursion is similarly derived as:

$$
\beta_{t-1}(y_{t-1}|x) = \sum_{y_t} \beta_t(y_t) \cdot M_t(y_t, y_{t-1}|x).
$$

(2.17)

The inference in CRF is completed by the posteriors derived as:

$$
P(y_t|x) = \frac{1}{Z(x)} \alpha_t(y_t|x) \cdot \beta_t(y_t|x),
$$

$$
P(y_t, y_{t-1}|x) = \frac{1}{Z(x)} \alpha_{t-1}(y_{t-1}|x) \cdot M_t(y_t, y_{t-1}|x) \cdot \beta_t(y_t|x).
$$

(2.18)

The learning of CRF is maximizing the conditional likelihood of the train data $D$ with respect to the CRF parameters $\lambda = \{ \lambda_{j,l} \}_{j,l}$ and $\eta = \{ \eta_{l,l'} \}_{l,l'}$. There is no analytical solution for this, however, the log-likelihood function,

$$
LL(\lambda, \eta; D) = \sum_{i=1}^{n} \left[ \sum_{t=1}^{T_i} \left( \sum_{j,l} \lambda_{j,l} \cdot I(y^i_t = l, f_j(x^i_t) = 1) + \sum_{l,l'} \eta_{l,l'} \cdot I(y^i_t = l, y^i_{t-1} = l') \right) - \log Z(x^i) \right],
$$

is concave in $\lambda$ and $\eta$ because of the log-convexity of the partition function. Thus the CRF learning is usually done by numerical optimization such as the Iterative Scaling algorithms [6, 8] or gradient search. Recently, it has been empirically shown that the conjugate gradient ascent or quasi-Newton variants (e.g., the memory efficient BFGS) often exhibit faster convergence than iterative scaling methods [50]. There are also many
other techniques proposed recently [25, 54, 64]. In the gradient-based CRF learning, the gradient of the log-likelihood is evaluated using the posteriors obtained from the inference, for instance,

$$\frac{\partial LL(\lambda, \eta; D)}{\partial \lambda_j} = \sum_{i=1}^{n} \left[ \sum_{t=1}^{T_i} \mathbb{I}(y_t^i = l, f_j(x_t^i) = 1) - \mathbb{E}_{P(y_t|x)} \left[ \sum_{t=1}^{T_i} \mathbb{I}(y_t = l, f_j(x_t^i) = 1) \right] \right].$$

(2.19)

The Viterbi decoding for CRF is similar to the HMM case. By defining $\delta_t(y_t = l) = \max_{y_1, \ldots, y_{t-1}} P(y_1, \ldots, y_{t-1}, y_t = l|x)$, the recursion is derived as:

$$\delta_{t+1}(y_{t+1} = l) = \max_{y_t} \delta_t(y_t) \cdot \frac{\beta_{t+1}(y_{t+1} = l|x) \cdot M_{t+1}(y_{t+1} = l, y_t)}{\beta_t(y_t|x)}.$$

(2.20)

[27] compared predictive performance of HMMs and CRFs (with corresponding HMM features) has been made for synthetic and real data. The synthetic experiment conducted in the paper is particularly interesting. In their setting, the true data generating process is modeled as a mixture of 1st- and 2nd-order transitions (i.e.,

$$P_a(y_t|y_{t-1}, y_{t-2}) = a \cdot P_2(y_t|y_{t-1}, y_{t-2}) + (1-a) \cdot P_1(y_t|y_{t-1})$$

), which makes the 1st-order dynamic HMMs and CRFs both suboptimal to the true structure. Here $a$ is a controllable parameter indicating the contribution of the 2nd-order transition effect. Thus $a = 0$ generates data with purely 1st-order transition (then we have a correct structure), while $a = 1$ generates extreme 2nd-order data. The scatter plot in Fig. 2.4 shows the test errors for two models. The points (data sets) away from the diagonal for a model indicate that the model is superior to the other model. From this result, one can conclude that CRF is more robust to HMM for the data with increasing 2nd-order transition effect. Considering the common modeling practice of approximating complex long-range dependencies by simpler structures, CRFs are more robust and appealing models than HMMs [27].

### 2.2 Why Discriminative Models?

From the above examples for classification, we see that discriminative models (Logistic Regression and CRFs) have unique merits compared to generative models (NB and
HMMs) in terms of prediction. Even though both models have similar graph structures and incorporate the same features, it has been empirically shown that the discriminative model has lower asymptotic errors, and is more robust to incorrect model structures than the generative model.

However, the generative model is often attractive in several aspects. One of its most notable merits is that we can synthesize or generate data samples. In addition, modeling decomposed local conditional distributions in directed generative models is often more intuitive than representing clique-based potential functions in undirected discriminative models. Moreover, the parameters of the local conditional distributions can give a better interpretation.

To retain the benefits of the generative model, one can consider optimizing a conditional distribution $P(y|x)$ inferred from the joint distribution of the generative model. It then reduces to learning the generative model with the conditional likelihood objective, instead of the joint likelihood. This approach is called the discriminative learning of generative models, and will be discussed further in Sec. 2.3 and Chap. 3. It can be seen as an alternative and implicit way of obtaining a discriminative model, and has been shown to yield significant improvement in prediction performance over the standard generative learning that maximizes the joint likelihood.

However, since the discriminative learning is performed with respect to the parameters of the generative model, it results in non-convex optimization in general. Hence
learning the model may suffer from computationally expensive numerical search and many local optima. On the other hand, as shown in the examples of the logistic regression and the CRF, the undirected discriminative model, by employing the log-linear clique potential functions, results in a convex objective\(^3\).

Focusing on the conditional distribution \(P(y|x)\) via either discriminative models or discriminative learning of generative models is crucial for accurate prediction. As the generative modeling/learning is based on fitting a model to data jointly, it would yield poor prediction performance if the data samples are insufficient to faithfully represent a joint distribution. This issue often arises in real problems, especially for high-dimensional data. I illustrate it on a simple regression task described below.

### 2.2.1 Example: Generative vs. Conditional Gaussian Mixture

We consider a simple regression problem with univariate \(x\) and \(y\). The data points are sampled from the following piecewise linear model:

\[
x \sim \text{Uniform}(0, 4),
\]

\[
y = \begin{cases} 
0.5x + 1 + \epsilon_1 & \text{if } x \leq 1 - \eta \text{ or } x \geq 3 + \eta \\
-0.5x + 3 + \epsilon_2 & \text{if } 1 + \eta \leq x \leq 3 - \eta \\
\epsilon_3 & \text{if } 1 - \eta < x < 1 + \eta \text{ or } 3 - \eta < x < 3 + \eta,
\end{cases}
\]

(2.21)

where \(\epsilon_1, \epsilon_2 \sim N(0, 0.01), \epsilon_3 \sim \text{Uniform}(1.5, 2.5)\), and \(\eta\) is a small constant (e.g., 0.1) which specifies the sizes of neutral zones adjoining the piecewise intervals. See Fig. 2.5(a) for the generated data samples (denoted by \(D_1\)) in a 2D plane.

A discriminative model that can represent the piecewise linearly correlated noisy data is a Gaussian conditional mixture (also known as the Mixture of Experts [21]). For the above data, the conditional mixture would have two linear Gaussian components

\(^3\)Here I assume that there are no hidden variables in the model.
Figure 2.5: Data sampled from the piecewise linear model Eq. (2.21). In (b), most data points at the central support around (2, 2) are eliminated.

(Experts) with log-quadratic\(^4\) logistic mixing (gating) functions:

\[
P_{cm}(y|x) = \sum_{m=1}^{2} \beta_m(x) \cdot N(y; a_m x + b_m, \nu_m), \quad \text{where} \tag{2.22}
\]

\[
\beta_1(x) = \frac{1}{1 + \exp(-0.5 q x^2 + r x + e)}, \quad \beta_2(x) = 1 - \beta_1(x).
\]

The parameters of the conditional mixture (cm) model \(P_{cm}(y|x)\) are denoted by \(\theta_{cm} = \{a_m, b_m, \nu_m\}_{m=1}^{2}, q, r, e\). Neglecting the (uniformly) random process around the neutral boundary zones (i.e., \(x \in (1 - \eta, 1 + \eta)\) and \(x \in (3 - \eta, 3 + \eta)\)), Eq. (2.21) can be simulated by the conditional mixture with:

\[
a_1 = 0.5, \quad b_1 = 1, \quad v_1 = 0.01,
\]

\[
a_2 = -0.5, \quad b_2 = 3, \quad v_2 = 0.01,
\]

\[
q = c, \quad r = 2c, \quad e = -1.5c, \tag{2.23}
\]

where \(c\) is sufficiently large to have sharp gating (i.e., \(\beta_m(x)\) close to either 0 or 1). See also Fig. 2.7(a) for the gating functions and prediction made by this model.

Alternatively, one can consider a generative model that jointly represents the data,

\(^4\)It is because the discontinuous pieces in the example data can be represented by quadratic gating.
namely, the Gaussian generative mixture (gm) defined on the stacked 2D data \([x, y]^\top\),

\[
P_{\text{gm}}(x, y) = \sum_{m=1}^{2} \alpha_m \cdot \mathcal{N} \left( \begin{bmatrix} x \\ y \end{bmatrix} ; \begin{bmatrix} \mu_x(m) \\ \mu_y(m) \end{bmatrix}, \begin{bmatrix} \sigma_{xx}(m) & \sigma_{xy}(m) \\ \sigma_{yx}(m) & \sigma_{yy}(m) \end{bmatrix} \right),
\] (2.24)

where \(\sigma_{xy}(m) = \sigma_{yx}(m)\) for \(m = 1, 2\), \(\alpha_2 = 1 - \alpha_1\), and \(0 \leq \alpha_1 \leq 1\).

I denote by \(\theta_{\text{gm}} = \{\{\mu_x(m), \mu_y(m), \sigma_{xx}(m), \sigma_{xy}(m), \sigma_{yy}(m)\}_{m=1}^{2}, \alpha_1\}\), the parameters of the generative model \(P_{\text{gm}}(x, y)\).

Although the intuition behind the generative mixture modeling is not as appealing in this case, it is not difficult to see that the family of distributions \(P_{\text{cm}}(y|x)\) realizable by all possible \(\theta_{\text{cm}}\) is equivalent to the family of conditional distributions \(P_{\text{gm}}(y|x)\) inferred from the generative mixture (i.e., \(P_{\text{gm}}(y|x) = P_{\text{gm}}(x, y)/P_{\text{gm}}(x)\)). In particular, one can find \(\theta_{\text{cm}}\) that corresponds to \(\theta_{\text{gm}}\) by the following equations:

\[
a_m = \sigma_{yx}(m) \cdot \sigma_{xx}^{-1}(m) \\
b_m = \mu_y(m) - \sigma_{yx}(m) \cdot \sigma_{xx}^{-1}(m) \cdot \mu_x(m) \\
v_m = \sigma_{yy}(m) - \sigma_{yx}(m) \cdot \sigma_{xx}^{-1}(m) \cdot \sigma_{xy}(m) \\
q = \sigma_{xx}^{-1}(2) - \sigma_{xx}^{-1}(1) \\
r = \sigma_{xx}^{-1}(2) \cdot \mu_x(2) - \sigma_{xx}^{-1}(1) \cdot \mu_x(1) \\
e = 0.5(\mu_x(1)^2 \cdot \sigma_{xx}^{-1}(1) - \mu_x(2)^2 \cdot \sigma_{xx}^{-1}(2)) - 0.5 \log \frac{\sigma_{xx}(2)}{\sigma_{xx}(1)} + \log \frac{\alpha_2}{\alpha_1}.\] (2.25)

While the mapping from \(\theta_{\text{gm}}\) to \(\theta_{\text{cm}}\) is unique, the mapping from \(\theta_{\text{cm}}\) to \(\theta_{\text{gm}}\) is one-to-many. This can be easily seen from the numbers of parameters (i.e., 11 parameters in \(\theta_{\text{gm}}\), but 9 in \(\theta_{\text{cm}}\)). For instance, one can arbitrarily change \(\sigma\) of \(\theta_{\text{gm}}\) (and change \(\mu\) accordingly) without altering the conditional distribution. Hence, in terms of \(P(y|x)\), the parameterization of the generative mixture is redundant.

Given train data \(D = \{(x^i, y^i)\}_{i=1}^{n}\), the standard maximum likelihood (ML) learning of the generative mixture, namely, “\(\max \sum_{i=1}^{n} \log P_{\text{gm}}(x^i, y^i)\)”, can be accomplished using the well-known EM algorithm which may yield a locally optimal solution. The conditional mixture learning, that is, “\(\max \sum_{i=1}^{n} \log P_{\text{cm}}(y^i|x^i)\)” is also non-convex due to the presence of the hidden mixture-component indicator variable, say \(z \in \{1, 2\}\).
Figure 2.6: The maximum-likelihood trained generative models on $D_1$ and $D_2$. 

Likewise, the learning resorts to iterative methods like the EM algorithm. More specifically, by letting $P(z = m|x) = \beta_m(x)$ and $P(y|z = m, x) = N(y; a_m x + b_m, v_m)$, the E-step computes $Q_i(z) = P(z|y^i, x^i)$ for all $i$, with respect to the current model $\theta_{cm}$, while in the M-step we maximize $\sum_{i=1}^{n} \sum_{z=1}^{2} Q_i(z) \cdot (\log P(z|x^i) + \log P(y^i|z, x^i))$. Note that the objective in the M-step is convex, however, the optimum does not admit a closed-form solution (e.g., the Newton-Raphson algorithm).

For both models, the EM learning is carried out starting from 10 random initial parameters (the same for both models using the conversion Eq. (2.25)), among which the best one is selected.

I consider two different train data sets: $D_1$ (Fig. 2.5(a)) is the original dataset containing sufficient data samples to represent the joint density faithfully, while $D_2$ (Fig. 2.5(b)) lacks samples at the center support around the point (2,2). The ML trained generative mixtures on $D_1$ and $D_2$ are denoted by $g_{m_1}$ and $g_{m_2}$, respectively. They are shown in Fig. 2.6, where each Gaussian component in the mixture is depicted by a contour ellipse. The figures also show the average joint log-likelihood (JLL) score, $\frac{1}{n} \sum_{i=1}^{n} \log P(x^i, y^i)$, and the conditional log-likelihood (CLL) score, $\frac{1}{n} \sum_{i=1}^{n} \log P(y^i|x^i)$, on both $D_1$ and $D_2$.

As $D_1$ has sufficient samples covering all the main supports of $P(x, y)$, $g_{m_1}$ produces a reasonable joint model that results in accurate prediction (Fig. 2.7(b)). For $D_2$, on
Figure 2.7: Prediction plots and gating functions: The dashed lines in the prediction plot correspond to the linear mean functions $y = a_m x + b_m$ of the conditional mixture. The green line indicates the final prediction which is selected from two linear mean functions by 0-1 thresholding of the gating function shown below. “True” in Fig 2.7(a) shows the conditional model in Eq. (2.23). For the generative mixture models, the equivalent conditional models are plotted using the conversion equations in Eq. (2.25).
Table 2.1: Prediction errors of generative and conditional mixtures.

<table>
<thead>
<tr>
<th></th>
<th>True</th>
<th>$g_{m1}$</th>
<th>$g_{m2}$</th>
<th>$c_{m1}$</th>
<th>$c_{m2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1210</td>
<td>0.1607</td>
<td>0.2561</td>
<td>0.1261</td>
<td>0.1237</td>
</tr>
</tbody>
</table>

the other hand, $g_{m1}$ is not optimal any more. Rather, the ML learning on $D_2$ preferably selects $g_{m2}$ (Fig. 2.6(b)) since it has a higher JLL score than that of $g_{m1}$ by ignoring the central support missed in $D_2$. However, even on $D_2$, $g_{m2}$ has a much lower CLL score ($-0.05$) than $0.30$ of $g_{m1}$, which consequently gives rise to the incorrect prediction model in Fig. 2.7(c).

The conditional mixtures $c_{m1}$ and $c_{m2}$ trained on $D_1$ and $D_2$, respectively, yield equally good prediction models as shown in Fig. 2.7(d) and Fig. 2.7(e). This can be explained from the fact that the discriminative model solely focuses on representing the functional relationship between input and output, instead of modeling the marginal input distribution, a task irrelevant for prediction. Table 2.1 summarizes prediction errors of generative mixtures and conditional mixtures on the two sets.

### 2.3 Discriminative Learning of Generative Models

In the previous sections, I have shown that discriminative models are more robust and accurate on prediction tasks than generative models in many situations. However, generative models are attractive in a number of data-driven modeling tasks. Among some of their advantages are the ability to easily incorporate domain knowledge, factorize complex problems into self-contained models, handle missing data and latent factors, and offer interpretability to results. Moreover, the discriminative models often suffer from the complexity in modeling the conditional distribution directly.

Some recent efforts have focused on the alternative direction of learning generative models discriminatively. The most popular discriminative learning method is the conditional likelihood maximization (CML). For the train data $D = \{(x^i, y^i)\}_{i=1}^n$, the CML
learning maximizes the conditional log-likelihood objective, namely,
\[
\arg\max_{\theta} \sum_{i=1}^{n} \log P(y^i|x^i, \theta),
\]
(2.26)
where \( \theta \) is the parameters of the generative model. The discriminative learning is shown to outperform the traditional maximum likelihood (ML) generative learning for a broad range of real data. It is also known that the discriminatively trained generative models focus on the decision boundaries while the generative learning concentrates on joint representation of data.

In the classification setting, the CML learning is known to achieve better classification performance than the traditional Maximum Likelihood (ML) fitting in a variety of situations [15, 41]. In automatic speech recognition, the discriminative parameter learning of HMMs and its benefit have been studied extensively [69].

However, unlike the case of the maximum likelihood estimator (MLE), the theoretical properties (e.g., asymptotic behavior, estimator bias) of the CML estimator have been mostly ignored and insufficiently studied. In addition, the CML learning is in general a non-convex optimization task in terms of the generative model parameters \( \theta \), hence suffering from non-unique solutions. Nevertheless, the discriminative learning is, in practice, very useful for more accurate prediction particularly when the structure of the model is incorrect.

In this section, I will describe a gradient-based optimization method for the CML learning. Although gradient-based optimization methods for specific static generative models (e.g., NB or its extension Tree Augmented Network (TAN) [15, 41]) have been proposed in the past, this work is a first to provide the derivation in a unifying framework (including structured classification and regression) for general generative models. The generative models are assumed to be directed graphical models (i.e., Bayesian Networks), possibly containing some hidden variables, all denoted by \( z \). By assuming that the local conditionals of the model are in the exponential family\(^5\), the complete joint distribution \( P(z, x, y) \) is also in the exponential family, which makes \( \nabla_{\theta} \log P(z, x, y|\theta) \) be

---
\(^5\)A probability distribution \( P(u; \theta) \) with the parameter vector \( \theta \in \mathbb{R}^s \) is said to belong to the exponential family if \( P(u; \theta) \propto \exp(\theta^\top \cdot T(u)) \), where \( T(u) \) is the feature vector (a function of data \( u \)) often called the sufficient statistics.
represented in a closed-form\(^6\). Furthermore, the exact inferences \(P(z|x, y)\) and \(P(y|x)\) are assumed to be tractable. The popular models such as NBs, TANs, Gaussians, HMMs, State-Space Models, and their mixtures satisfy the assumptions.

### 2.3.1 Conditional Likelihood Maximization

The CML learning tries to find a parameter vector \(\theta\) (among the parameter space \(\Theta\) of the generative model; i.e, \(\theta \in \Theta\)) that maximizes the conditional likelihood. For the train data \(D = \{(x^i, y^i)\}_{i=1}^n\), the conditional log-likelihood objective \(CLL\) is defined as:

\[
CLL(\theta; D) = \sum_{i=1}^n \log P(y^i|x^i, \theta) = \sum_{i=1}^n \left[ \log P(y^i|x^i, \theta) - \log P(x^i|\theta) \right].
\]

The \(CLL\) objective is directly related with the prediction. Moreover, the generative model implicitly defines a family of conditional distributions,

\[
C_{\Theta} = \{P(y|x, \theta) = \frac{P(x, y|\theta)}{\int_y P(x, y|\theta)} : \theta \in \Theta\}.
\]

The induced family \(C_{\Theta}\) is as powerful as that of the discriminative model with similar structure/features. In particular, it is easy to see that the generative models in the previous sections have the families of conditional distributions equivalent to the corresponding discriminative models. For example, the conditional distribution inferred from the NB can be written as:

\[
P(y = 1|x) = \left[ 1 + \exp \left( \log \left( \frac{1 - \pi}{\pi} \right) + \sum_{j=1}^p \left[ x_j \cdot \log \left( \frac{\mu_{j,1}}{\mu_{j,0}} \right) + (1 - x_j) \cdot \log \left( \frac{1 - \mu_{j,1}}{1 - \mu_{j,0}} \right) \right] \right) \right]^{-1}.
\]

In fact, the logistic regression can be seen as a reparameterization of the NB, namely,

\[
w_j = -\log \left( \frac{\mu_{j,1}}{\mu_{j,0}} \right) + \log \left( \frac{1 - \mu_{j,1}}{1 - \mu_{j,0}} \right), \quad \text{for} \quad j = 1, \ldots, p,
\]

\[
b = -\log \left( \frac{1 - \pi}{\pi} \right) - \sum_{j=1}^p \log \left( \frac{1 - \mu_{j,1}}{1 - \mu_{j,0}} \right).
\]

(2.29)

Noticing that the \(CLL\) objective is a natural maximum likelihood estimator for the discriminative model, the CML learning does this for the induced family of conditional

\(^6\)Unlike the undirected graphical models, this is a unique property of directed graphical models where the partition functions have simple forms.
distributions from the generative model. In this sense, the CML learning of the generative model can be regarded as an implicit way to realize the discriminative model. In the above correspondence, unlike the strictly concave objective of the logistic regression, the CML solution for the NB is not unique since there exists a many-to-one mapping from the NB parameters to the logistic regression parameters. In general, the CLL objective of the generative model has many global and local optima. Thus when compared to the discriminative model, the discriminative learning of the generative model takes the benefits of generative models and superior prediction performance to the generative ML learning at the expense of non-convex optimization.

The CLL objective can be locally optimized using a parametric gradient search. In [15, 41], a derivation for the CLL gradient for NB and TAN models in static classification has been introduced. Here I provide a unifying way to evaluate the parametric gradient of the CLL for general generative models. From Eq. (2.27), the gradient of the CLL with respect to $\theta$ is given as:

$$\frac{\partial \text{CLL}(\theta; D)}{\partial \theta} = \sum_{i=1}^{n} \left[ \frac{\partial}{\partial \theta} \log P(x^i, y^i|\theta) - \frac{\partial}{\partial \theta} \log P(x^i|\theta) \right].$$  \hspace{1cm} (2.30)

The first term $\frac{\partial}{\partial \theta} \log P(x, y|\theta)$, the gradient of the joint log-likelihood, is also known as the joint Fisher score. Assuming that all the conditional densities in the BN belong to the exponential families, it can be evaluated straightforwardly if $P(x, y)$ has no hidden variables. The presence of hidden variables $z$, on the other hand, trivially results in the expectation of the gradient of the complete log-likelihood (including $z$), namely,

$$\frac{\partial}{\partial \theta} \log P(x, y|\theta) = \frac{1}{P(x, y|\theta)} \cdot \int_z P(z, x, y|\theta) \cdot \frac{\partial}{\partial \theta} \log P(z, x, y|\theta)$$

$$= \int_z P(z|x, y, \theta) \cdot \frac{\partial}{\partial \theta} \log P(z, x, y|\theta)$$

$$= \mathbb{E}_{P(z|x, y, \theta)} \left[ \frac{\partial}{\partial \theta} \log P(z, x, y|\theta) \right].$$  \hspace{1cm} (2.31)

The expectation can be computed easily as long as the inference for $P(z|x, y)$ is tractable. The second term $\frac{\partial}{\partial \theta} \log P(x|\theta)$ of Eq. (2.30), the derivative of the measurement or input log-likelihood, is the expectation (with respect to $y$) of the joint
log-likelihood in the same manner by treating the target \( y \) as hidden variables. That is,
\[
\frac{\partial}{\partial \theta} \log P(x|\theta) = \mathbb{E}_{P(y|x, \theta)} \left[ \frac{\partial}{\partial \theta} \log P(x, y|\theta) \right].
\] (2.32)

From Eq. (2.31) and Eq. (2.32), the gradient of the \( \text{CLL} \) is then represented as:
\[
\frac{\partial \text{CLL}(\theta; D)}{\partial \theta} = \sum_{i=1}^{n} \mathbb{E}_{P(z|x^i,y^i)} \left[ \frac{\partial}{\partial \theta} \log P(z, x^i, y^i|\theta) \right] - \sum_{i=1}^{n} \mathbb{E}_{P(z,y|x^i)} \left[ \frac{\partial}{\partial \theta} \log P(z, x^i, y|\theta) \right], \quad (z \text{ present}),
\] (2.33)
\[
\frac{\partial \text{CLL}(\theta; D)}{\partial \theta} = \sum_{i=1}^{n} \frac{\partial}{\partial \theta} \log P(x^i, y^i|\theta) - \sum_{i=1}^{n} \mathbb{E}_{P(y|x^i)} \left[ \frac{\partial}{\partial \theta} \log P(x^i, y|\theta) \right], \quad (z \text{ absent}).
\] (2.34)

The generative learning tries to make the gradient of the joint log-likelihood vanished. That is, the MLE satisfies \( \sum_{i=1}^{n} \frac{\partial}{\partial \theta} \log P(x^i, y^i|\theta) = 0 \). This means the sum of the Fisher scores on the train data has to be 0, which corresponds to fitting the model globally to the data. On the other hand, the CML learning tries to make the gradient of the \( \text{CLL} \) become 0, which is equivalent to minimizing the difference between the sum of the Fisher scores on data and the sum of the expected Fisher scores with respect to the model. Hence the CML learning focuses on the target posterior \( P(y|x) \) of the model, making it as close to the empirical conditional as possible.

### 2.3.2 Example: Classification with Mixtures of Gaussians

We now see a concrete example that compares the generative (ML) learning and the discriminative (CML) learning on the mixtures of 2D Gaussians in static classification. The generative model used for binary classification has a mixture of two Gaussians for each class. The model is composed of the class variable \( y \) taking either 1 or 2, the input measurement in a 2D plane \( x \in \mathbb{R}^2 \), and the hidden variable \( z \in \{1, 2\} \) indicating a particular component of the mixtures. The joint likelihood of the model is written as \( P(x, y) = P(y) \cdot P(x|y) = P(y) \cdot \sum_{z=1}^{2} P(z|y) \cdot P(x|z, y) \). For simplicity, the class prior and the mixing proportions are assumed to be known as \( P(y = 1) = 0.5 \) and \( P(z = 1|y = 1) = P(z = 1|y = 2) = 0.5 \). The class conditional is a mixture of two
In Eq. (2.33), the parametric gradient of the \( \text{CLL} \) variable \( z \) with respect to the current model, \( \mu_{y,z} \), \( \Sigma_{y,z} \) is a set of \((d \times d)\) symmetric positive definite matrices. The complete log-likelihood is given as:

\[
\log P(z, x, y) = \log P(y) + \log P(z | y) + \log P(x | z, y) = \log(0.25) - \frac{1}{2} \log |\Sigma_{y,z}| - \frac{1}{2}(x - \mu_{y,z})^\top \Sigma_{y,z}^{-1}(x - \mu_{y,z}).
\]

Given the train data \( \{(x^i, y^i)\}_{i=1}^n \), the generative learning maximizes \( \sum_{i=1}^n \log P(x^i, y^i) \).

In the EM learning, the E-step computes the posterior distributions of the hidden variable \( z \) with respect to the current model,

\[
Q_i(z) = P(z | x^i, y^i) = \frac{P(z, x^i, y^i)}{\sum_{z'} P(z', x^i, y^i)} \quad \text{for} \quad i = 1, \ldots, n.
\]

The M-step maximizes \( \sum_{i=1}^n \sum_{z=1}^2 Q_i(z) \cdot \log P(z, x^i, y^i) \), or equivalently, solves the equation of the sum of the expected Fisher scores being 0, namely,

\[
\sum_{i=1}^n \sum_{z=1}^2 Q_i(z) \cdot \frac{\partial}{\partial \theta} \log P(z, x^i, y^i) = 0.
\]

It is easy to see that the Fisher score can be evaluated in a closed form:

\[
\frac{\partial}{\partial \theta} \log P(z, x, y) = \begin{cases} 
\Sigma_{y,z}^{-1}(x - \mu_{y,z}) & \text{if } \theta = \mu_{y,z} \\
\frac{1}{2} \Sigma_{y,z} - \frac{1}{2}(x - \mu_{y,z})(x - \mu_{y,z})^\top & \text{if } \theta = \Sigma_{y,z}^{-1} \\
0 & \text{otherwise}
\end{cases}
\]

In this way, the M-step has a closed-form solution, for instance,

\[
\mu_{1,1}^{\text{new}} = \frac{\sum_{i:y^i=1} Q_i(1) \cdot x^i}{\sum_{i:y^i=1} Q_i(1)}, \quad \Sigma_{1,1}^{\text{new}} = \frac{\sum_{i:y^i=1} Q_i(1) \cdot (x^i - \mu_{1,1}^{\text{new}})(x^i - \mu_{1,1}^{\text{new}})^\top}{\sum_{i:y^i=1} Q_i(1)}.
\]

The CML learning maximizes \( \text{CLL} = \sum_{i=1}^n \log P(y^i | x^i) \). Following the derivation in Eq. (2.33), the parametric gradient of the \( \text{CLL} \) objective is:

\[
\frac{\partial \text{CLL}}{\partial \theta} = \sum_{i=1}^n \mathbb{E}_{P(x^i, y^i)} \left[ \frac{\partial}{\partial \theta} \log P(z, x^i, y^i) \right] - \sum_{i=1}^n \mathbb{E}_{P(y^i | x^i)} \left[ \frac{\partial}{\partial \theta} \log P(z, x^i, y) \right].
\]
The second posterior \( R_i(y, z) = P(y, z|x^i) \) can be computed easily by:

\[
R_i(y, z) = \frac{P(z, x^i, y)}{\sum_{y', z'} P(z', x^i, y')}
\]

In this way, the CLLL gradient is:

\[
\frac{\partial \text{CLL}}{\partial \mu_{y,z}} = \sum_{i : y^i = y} Q_i(z) \cdot \Sigma_{y,z}^{-1}(x^i - \mu_{y,z}) - \sum_{i=1}^n R_i(y, z) \cdot \Sigma_{y,z}^{-1}(x^i - \mu_{y,z})
\]

\[
\frac{\partial \text{CLL}}{\partial \Sigma_{y,z}} = \sum_{i : y^i = y} Q_i(z) \cdot \left[ \frac{1}{2} \Sigma_{y,z} - \frac{1}{2} (x^i - \mu_{y,z})(x^i - \mu_{y,z})^\top \right] - \sum_{i=1}^n R_i(y, z) \cdot \left[ \frac{1}{2} \Sigma_{y,z} - \frac{1}{2} (x^i - \mu_{y,z})(x^i - \mu_{y,z})^\top \right].
\]

Note that for \( \theta = \Sigma_{y,z}^{-1} \), a special care needs to be taken to guarantee symmetric positive definiteness during the gradient ascent updates. A simple way is to do a Cholesky-like reparameterization \( \Sigma_{y,z}^{-1} = A^\top A \), which results in \( \frac{\partial \text{CLL}}{\partial A} = 2A \cdot \frac{\partial \text{CLL}}{\partial \Sigma_{y,z}} \) by the chain rule.

To compare the prediction performance of two learning methods, a synthetic experiment is conducted. The data is sampled from 8 2D Gaussians with unit spherical covariances, 4 of them generating the class \( y = 1 \) while the other 4 Gaussians for the class \( y = 2 \), as shown in Fig. 2.8(a). The number of samples is chosen large (about 1,000) to see the asymptotic behavior of the learned models. The mixture model (2 components for each class) discussed above is obviously suboptimal to the true data generating process. However, for a certain parameter vector, the classification error becomes almost 0 as depicted in Fig. 2.8(b). In fact, the model in Fig. 2.8(b) is one of the global maxima for the CML learning (CLL = −0.0014). Note that this is not the global maxima for the generative learning (\( JLL = -6738.04 \)) since the model in Fig. 2.8(c) records \( JLL = -6314.23 \). The generative learning can find a model in Fig. 2.8(c) (depending on the initial parameter choice), however, as the CLL score indicates (−591.97), the classification error is about 0.5, no better than random guess.

Another important issue is that the learned models sensitively depend on the initial model. This is due to the non-convex objectives and the iterative nature of the EM algorithm (for the ML learning) and the gradient-based optimization (for the CML learning). For each row (from the second to the last) in Fig. 2.8, the leftmost plot shows the initial model randomly chosen. The CML learning reaches almost 0 asymptotic
Figure 2.8: Asymptotic behavior of generative and discriminative learning.
error rate for the first two initial choices, despite different decision boundaries from the intuitive one in Fig. 2.8(b). The generative learning yields much worse prediction even though it achieves higher (joint) likelihood scores. However, for the last initial model, the discriminative learning fails reaching a global optimum, with the lower CLL score and the higher error rate. Even in this case, the CML learning results in lower prediction error than the ML learning.

2.3.3 Evaluation on Real Data Classification

In order to demonstrate the generalization performance of the discriminative learning approach, the evaluation is conducted on the real data for the problems of structured output classification. The evaluation for the structured output regression will be discussed in Chap. 3 since the CML learning for dynamical systems (the conventional generative models for the continuous multivariate state sequence) is novel and worth to be discussed separately.

As discussed in Sec. 2.1.2, the HMM is used as a generative model. Recall that the state variables of HMMs are no more hidden variables, but the target outputs. The datasets are from the computer vision (Kiosk speaker detection dataset) and the information extraction (FAQ dataset). In the Kiosk, given the sequence of the measurements (binary features from the face detectors and the audio cues), the goal is to label each time frame to 1 or 2, indicating whether the human is speaking or not, respectively. For 5 labeled sequences of length around 2,000, the leave-1-out test is performed. The FAQ dataset is first used in the information extraction community [35]. For the FAQ document with a particular topic (e.g., BSD or Fetish), each sentence in a document is labeled to one of the tags, \{ head, question, answer, tail \}. The measurements are 24 binary features extracted from the sentences (e.g., Start with a capital letter?, Contain a numeric letter?, or Length less than 3?).

Table 2.2 shows the test accuracies from the leave-1-out cross validation. The CML learning yields more accurate prediction than the ML. Even compared with the discriminative CRF, the CML learning of generative models is never inferior.

In the next two chapters, I tackle the structured output regression problem using
Table 2.2: Structured output classification test accuracies (%).

<table>
<thead>
<tr>
<th></th>
<th>Kiosk</th>
<th>FAQ-BS</th>
<th>FAQ-Fetish</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM-ML</td>
<td>90.14 ± 3.86</td>
<td>92.48 ± 4.10</td>
<td>92.86 ± 7.91</td>
</tr>
<tr>
<td>HMM-CML</td>
<td>95.94 ± 1.73</td>
<td>97.86 ± 3.46</td>
<td>97.76 ± 2.49</td>
</tr>
<tr>
<td>CRF</td>
<td>94.27 ± 2.19</td>
<td>97.60 ± 2.16</td>
<td>98.66 ± 0.92</td>
</tr>
</tbody>
</table>

the discriminative approaches discussed thus far. The widely used generative model for this purpose is the state-space model, sometimes called the linear dynamical system. I consider the discriminative learning of the state-space model as well as develop a novel discriminative model that corresponds to the state-space model.
Chapter 3

Discriminative Learning of Dynamical Systems

I consider the problem of dynamical state estimation of time-series sequences. The problem can be formulated as a structured output regression problem, estimating a continuous multivariate state sequence, $y = y_1 \cdots y_T$, from the measurement sequence, $x = x_1 \cdots x_T$, where $y_t \in \mathbb{R}^d$ and $x_t \in \mathbb{R}^p$. Its applications typically in computer vision include 3D human motion and body pose estimation for moving objects from sequences of images.

A problem resembling the dynamical state estimation, when $y_t$ is a discrete label instead of continuous multivariate, is known as the structured output classification. The most popular generative model in this realm is the Hidden Markov Model (HMM). Traditional Maximum Likelihood (ML) learning of generative models such as HMMs is not directly compatible with the ultimate goal of label prediction (namely, $y$ given $x$), as it optimizes the fit of the models to data jointly, $x$ and $y$. Recently, discriminative models such as Conditional Random Fields (CRFs) and Maximum Entropy Markov Models (MEMMs) were introduced to address the label prediction problem directly, resulting in superior performance to the generative models [27, 35].

Despite a broad success of discriminative models in the discrete state domain, the use of discriminative dynamic models for continuous multivariate state estimation is not widespread. One reason for this is that a natural reparameterization-based transformation of generative dynamical systems to conditional models may violate density integrability constraints. For example, a naive extension of the state-space model (SSM) to CRF imposes nontrivial constraints on the CRF parameters to ensure finiteness of the log-partition function, making convex or general gradient-based optimization complex and prone to numerical failure.
As an alternative to CRF-based models in continuous state sequence domains I propose to learn generative dynamic models discriminatively. This approach has been well studied in classification settings: learning generative models such as Tree-Augmented Naive Bayes (TAN) or HMMs discriminatively via maximizing the conditional likelihood yields better prediction performance than the traditional maximum likelihood estimator [15, 19, 23, 37, 41]. The main contribution of this chapter is to extend the discriminative learning approach to dynamic models and the structured output regression problem. Namely, we learn dynamic models that directly optimize the accuracy of the output prediction rather than jointly increasing the likelihood of input and output.

I introduce two discriminative learning algorithms for the generative probabilistic dynamical system, \( P(x, y) \). One is to maximize the conditional log-likelihood of the entire state sequence \( y \), that is, arg \( \max \log P(y|x) \), while the other is for the individual state slices \( y_t \), namely, arg \( \max(1/T) \sum_{t=1}^{T} \log P(y_t|x) \). These objectives are not convex in general, however, the gradient-based optimization yields superior prediction performance to that of the standard ML learning. In addition, I derive computationally efficient methods for gradient evaluation for two objectives.

For several human motion datasets, we compare the prediction performance of the competing models including nonlinear and latent variable dynamic models. The discriminative learning algorithms on the SSM can provide significantly lower prediction error than the standard maximum likelihood estimator, often comparable to estimates of computationally more expensive and parameter sensitive nonlinear or latent variable models. Thus the discriminatively trained SSM offers a highly desired combination of high estimation accuracy and low computational complexity.

The chapter is organized as follows: In the next section the SSM is briefly reviewed. In Sec. 3.2, I discuss several known issues related to directly applying discriminative models to the continuous multivariate state domain. Then in Sec. 3.3, the proposed discriminative learning algorithms for the SSM are described, followed by how they can be extended to nonlinear models. After reviewing related work in Sec. 3.4, the evaluation on the motion data is provided in Sec. 3.5.
3.1 State-Space Model

The SSM, often called the Linear Dynamical System (LDS), is a generative sequence model with a graphical representation shown in Fig. 3.1(a). The SSM assumes transition and emission densities to be linear Gaussian. Its joint distribution can be factorized into two parts; the Auto-Regressive (AR) model and the product of Gaussians, namely,

\[ P(x, y) = P_{AR}(y) \cdot P_{Gauss}(x|y), \]

where

\[ P_{AR}(y) = P(y_1) \cdot \prod_{t=2}^{T} P(y_t|y_{t-1}) = \mathcal{N}(y_1; m_0, V_0) \cdot \prod_{t=2}^{T} \mathcal{N}(y_t; A y_{t-1}, \Gamma), \]

\[ P_{Gauss}(x|y) = \prod_{t=1}^{T} P(x_t|y_t) = \prod_{t=1}^{T} \mathcal{N}(x_t; Cy_t, \Sigma). \]  

(3.1)

Here \( \theta = \{m_0, V_0, A, \Gamma, C, \Sigma\} \) is the SSM parameter set. The joint log-likelihood, \( LL = \log P(x, y) \)\(^1\) is (up to a constant):

\[
LL = -\frac{1}{2} \left[ (y_1 - m_0)^\top V_0^{-1} (y_1 - m_0) + \log |V_0| + \sum_{t=2}^{T} (y_t - Ay_{t-1})^\top \Gamma^{-1} (y_t - Ay_{t-1}) + (T - 1) \log |\Gamma| + \sum_{t=1}^{T} (x_t - Cy_t)^\top \Sigma^{-1} (x_t - Cy_t) + T \log |\Sigma| \right].
\]

(3.2)

The task of inference is to compute the filtered state densities, \( P(y_t|x_1, \ldots, x_t) \) and the smoothed densities, \( P(y_t|x) \). The linear Gaussian assumption of the SSM implies Gaussian posteriors that can be evaluated in linear time using the well-known Kalman filtering or RTS smoothing methods. I denote the means and the covariances of these

\(^1\) For brevity, I will often drop the dependency on \( \theta \) in the notation.
posterior densities by:

\[ \hat{m}_t := \mathbb{E}[y_t|x_1, \ldots, x_t], \ \hat{V}_t := \mathbb{V}(y_t|x_1, \ldots, x_t), \]

\[ m_t := \mathbb{E}[y_t|x], \ V_t := \mathbb{V}(y_t|x), \ \Sigma_{t,t-1} := \text{Cov}(y_t, y_{t-1}|x). \quad (3.3) \]

To learn the SSM, one needs to find \( \mathbf{\theta}_{ssm} \) that optimizes a desired objective function. In the supervised setting that we assume throughout the paper, for the given train data \( D = \{(x^i, y^i)\}_{i=1}^n \), the generative learning maximizes the joint log-likelihood, \( \sum_{i=1}^n LL(x^i, y^i) \), which has a closed-form solution by setting the gradients,

\[
\begin{align*}
\frac{\partial LL}{\partial m_0} &= V_0^{-1}(y_1 - m_0), \\
\frac{\partial LL}{\partial V_0^{-1}} &= \frac{1}{2} V_0 - \frac{1}{2} (y_1 - m_0)(y_1 - m_0)^\top, \\
\frac{\partial LL}{\partial A} &= \Gamma^{-1} \cdot \sum_{t=2}^T (y_t y_{t-1}^\top - A y_{t-1} y_{t-1}^\top), \\
\frac{\partial LL}{\partial \Gamma^{-1}} &= \frac{T-1}{2} \Gamma - \frac{1}{2} \sum_{t=2}^T (y_t - Ay_{t-1})(y_t - Ay_{t-1})^\top, \\
\frac{\partial LL}{\partial C} &= \Sigma^{-1} \cdot \sum_{t=1}^T (x_t y_t^\top - Cy_t y_t^\top), \\
\frac{\partial LL}{\partial \Sigma^{-1}} &= \frac{T}{2} \Sigma - \frac{1}{2} \sum_{t=1}^T (x_t - Cy_t)(x_t - Cy_t)^\top, \quad (3.4)
\end{align*}
\]

to 0. For instance, the emission matrix,

\[ C^* = \left[ \sum_{i=1}^n \sum_{t=1}^{T_i} x_i^t y_i^t \right]^{-1} \left[ \sum_{i=1}^n \sum_{t=1}^{T_i} y_i^t y_i^t \right]^{-1}, \]

where \( T_i \) is the length of the \( i \)-th sequence.

The ML learning of the generative model is intended to fit the model to data jointly on \( x \) and \( y \). However, in tracking we are often more interested in finding a model that yields a high accuracy of predicting \( y \) from \( x \), an objective not achieved by ML learning in general. It is therefore tempting to employ discriminative models which explicitly focus on the desired goal. In the discrete state domain, CRFs and MEMMs are such models shown to outperform the generative models like HMMs. Unfortunately, as discussed in the next section, developing CRF- or MEMM-like discriminative models in the continuous multivariate state domain can be a challenge.
3.2 Difficulty in Discriminative Dynamic Modeling

3.2.1 Conditional Random Fields

The CRF models the conditional probability of \( y \) given \( x \). Since \( P(y|x) \propto P(x, y) \), the log-conditional log \( P(y|x) \) has the same form as Eq. (3.2) except that those terms that are not involved in \( y \) (e.g., \( x_t^\top \Sigma^{-1}x_t \)) can be removed as they will be marginalized out into the log-partition function. One can naively transform \( \theta_{ssm} \) into the CRF parameters \( \theta_{crf} = \{ \Lambda_b, \Lambda_1, \Lambda, \Lambda_T, \Lambda_A, \Lambda_C \} \) as follows:

\[
\begin{align*}
\Lambda_b &= V_0^{-1}m_0, \\
\Lambda_A &= \Gamma^{-1}A, \\
\Lambda_1 &= -\frac{1}{2}(V_0^{-1} + A^\top \Gamma^{-1}A + C' \Sigma^{-1}C), \\
\Lambda &= -\frac{1}{2}(\Gamma^{-1} + A^\top \Gamma^{-1}A + C' \Sigma^{-1}C), \\
\Lambda_T &= -\frac{1}{2}(\Gamma^{-1} + C' \Sigma^{-1}C). \\
\end{align*}
\]

(3.5)

Then the SSM-counterpart CRF model can be written as:

\[
P(y|x, \theta_{crf}) = \frac{\exp(\Phi(x, y; \theta_{crf}))}{Z(x; \theta_{crf})}, \text{ where}
\]

\[
\Phi(x, y; \theta_{crf}) = \Lambda_b^\top y_1 + y_1^\top \Lambda_1 y_1 + \sum_{t=2}^{T-1} y_t^\top \Lambda y_t + y_T^\top \Lambda_T y_T + \sum_{t=2}^{T} y_t^\top \Lambda_A y_{t-1} + \sum_{t=1}^{T} x_t^\top \Lambda_C y_t,
\]

\[
Z(x; \theta_{crf}) = \int_y \exp(\Phi(x, y; \theta_{crf})).
\]

(3.6)

Note that \( \Lambda_b \in \mathbb{R}^{d \times 1}, \Lambda_1, \Lambda, \Lambda_T, \Lambda_A \in \mathbb{R}^{d \times d}, \) and \( \Lambda_C \in \mathbb{R}^{p \times d} \). Below, I abuse the notation by defining \( \Lambda_t := \Lambda \) for \( 2 \leq t \leq T - 1 \), which enables us to compactly represent \( \Lambda_1, \Lambda, \) and \( \Lambda_T \) all together as \( \Lambda_t \) for \( 1 \leq t \leq T \).

The (conditional) log-likelihood log \( P(y|x, \theta_{crf}) \) is concave in \( \theta_{crf} \) because both \( \Phi(x, y; \theta_{crf}) \) and the log-partition function log \( Z(x; \theta_{crf}) \) are convex. However, the reparameterization produces extra constraints on the CRF parameter space. The thorough set of constraints is not immediately obvious and includes constraints such as the symmetry and negative definiteness of \( \Lambda_t \). Other constraints can be revealed during the inference phase.
In the assumed chain-structured CRF as shown in Fig. 3.1(b), the potential function $M_t(\cdot)$ defined on the clique at time $t$ can be denoted as:

$$M_1(y_1|x) = e^{y_1^\top \Lambda_1 y_1 + x_1^\top \Lambda_C y_1},$$

$$M_t(y_t, y_{t-1}|x) = e^{y_t^\top \Lambda_t y_t + y_{t-1}^\top \Lambda_A y_{t-1} + x_t^\top \Lambda_C y_t}, \quad t \geq 2. \quad (3.7)$$

With the initial condition, $\alpha_1(y_1|x) = M_1(y_1|x)$, the forward message is defined recursively (for $t \geq 2$) as,

$$\alpha_t(y_t|x) = \int_{y_{t-1}} \alpha_{t-1}(y_{t-1}|x) \cdot M_t(y_t, y_{t-1}|x). \quad (3.8)$$

Since $\alpha_t(y_t|x)$ is an unnormalized Gaussian, it can be represented by a triple $(r_t, P_t, q_t) \in (\mathbb{R}, \mathbb{R}^{d \times d}, \mathbb{R}^d)$, where $\alpha_t(y_t|x) = r_t \exp(y_t^\top P_t y_t + q_t^\top y_t)$. For a feasible $\theta_{crf}$,

$$r_t = r_{t-1} \left| -\pi P_{t-1}^{-1} \right|^{1/2} \exp(-\frac{1}{4} q_{t-1}^\top P_{t-1}^{-1} q_{t-1}),$$

$$q_t = \Lambda_C^\top x_t - \frac{1}{2} \Lambda_A P_{t-1}^{-1} q_{t-1}, \quad \text{for} \quad 2 \leq t \leq T, \quad \text{and}$$

$$P_t = \Lambda - \frac{1}{4} \Lambda_A P_{t-1}^{-1} \Lambda_A^\top, \quad \text{for} \quad 2 \leq t \leq T - 1, \quad (3.9)$$

with the boundary conditions: $r_1 = 1, P_1 = \Lambda_1, q_1 = \Lambda_b + \Lambda_C^\top x_1$, and $P_T = \Lambda_T - \frac{1}{4} \Lambda_A P_{T-1}^{-1} \Lambda_A^\top$. Because $Z(x) = \int_{y_T} \alpha_T(y_T|x)$, $P_t$, for all $t$, must be negative definite to guarantee not only a proper (integrable) density with a finite log-partition function, but also proper forward messages $\alpha_t(\cdot)$. As shown in the recursion in Eq. (3.9), however, these conditions may produce nontrivial constraints on $\theta_{crf}$.

The backward recursion, that can be similarly derived, adds additional constraints on the parameters. As a result, specifying the feasible parameter space of continuous conditional dynamic models is difficult. I will address this issue in Chap. 4.

### 3.2.2 Maximum Entropy Markov Models

MEMM has a graphical structure depicted in Fig. 3.1(c). Despite the well-known label bias problem, its simple learning procedure that does not require forward/backward recursion is very attractive. Given a complete data ($x, y$), the likelihood function can be factored into terms related with individual slices ($y_{t-1}, x_t, y_t$), and subsequently treated as a set of independent slice instances. Learning MEMM is equivalent to training
a static classifier or regression function $P(y_t|y_{t-1},x_t)$ for the i.i.d. data with the output $y_t$ and the input $(y_{t-1}, x_t)$.

MEMM with the linear Gaussian conditional, namely,

$$y_t|y_{t-1}, x_t \sim \mathcal{N}(y_t; A_y y_{t-1} + A_x x_t + e, W),$$

(3.10)
can be seen as a counterpart of the SSM. The prediction is done by the recursion,

$$P(y_t|x) = \int_{y_{t-1}} P(y_t|y_{t-1}, x_t) \cdot P(y_{t-1}|x).$$

Note that in MEMMs the smoothed posterior $P(y_t|x)$ equals the filtered posterior $P(y_t|x_1, ..., x_t)$, effectively removing the influence of future measurements on estimating the current state. The mean estimate $m_t = \mathbb{E}[y_t|x]$ is:

$$m_t = A_y m_{t-1} + A_x x_t + e.$$  

(3.11)

Eq. (3.11) points to another deficiency of linear MEMMs. The next state estimate is linearly related with the previous state mean, where the coefficient $A_y$ is determined by the multivariate linear regression learning with data treated slice-wise independently. If the learned $A_y$ is unstable\(^2\), the state estimates become unbounded. As a result, the state estimation error can be significantly amplified in this MEMM setting.

This behavior may be reduced when non-linear or non-Gaussian noise models are used. In [53], for instance, a complex nonlinear regression function (Bayesian Mixture of Experts) was applied to the 3D human body pose estimation problem. However, the failure of simple linear MEMM points to prevalent role of local functions over the MEMM’s overall discriminative model structure. In other words, the success of MEMM may be strongly dependent on the performance of the employed static regression functions.

### 3.3 Discriminative Learning of State-Space Models

The analysis of traditional conditional dynamic models points to possible modes of failure when such models are applied to continuous state domains. To address these deficiencies I suggest to learn the generative SSM with discriminative cost functions.

\(^2\)Eigenvalues of matrix $A$ have absolute magnitudes exceeding 1.
As the discriminative learning of HMM has shown to outperform generative learning in structured classification settings, the same approach can be brought to benefit the task of structured output regression in continuous domains. I propose two discriminative objectives to solve the problem of discriminative learning of the SSM. The optimal parameter estimation is accomplished by an efficient gradient search on the two objectives. I also show how the discriminative learning can be extended to a general family of nonlinear dynamic models.

### 3.3.1 Conditional Likelihood Maximization (CML)

The goal of the CML learning is to find the SSM parameters that maximize the conditional likelihood of $y$ given $x$, an objective directly related to the goal of accurate state prediction. The conditional log-likelihood objective for the data $(x, y)$ is defined as:

$$
CLL = \log P(y|x) = \log P(x, y) - \log P(x).
$$

The $CLL$ objective is, in general, non-convex in the model parameter space. However, the objective can be locally optimized using a general gradient search. The gradient of $CLL$ with respect to $\theta_{ssm}$ is:

$$
\frac{\partial CLL}{\partial \theta_{ssm}} = \frac{\partial \log P(x, y)}{\partial \theta_{ssm}} - \frac{\partial \log P(x)}{\partial \theta_{ssm}}.
$$

The first term, the gradient of the complete log-likelihood (the Fisher score) is shown in Eq. (3.4). The second term, the gradient of the observation log-likelihood, is essentially the expected Fisher score by the model given $x$ only. Because the Fisher score, as shown in Eq. (3.4), is a sum of the 2nd-order moments (i.e., those related with $y_t y_t^\top, y_t y_{t-1}^\top$, or $y_t$), the expected Fisher score can be easily computed once we have the
posterior $P(y|x)$. For example, using the fact that $E[XY^\top] = E[X]E[Y]^\top + \text{Cov}(X,Y)$, the gradient with respect to the transition covariance is:

$$
\frac{\partial \log P(x)}{\partial \Gamma^{-1}} = E_{P(y|x)} \left[ \frac{\partial LL}{\partial \Gamma^{-1}} \right] = \frac{T-1}{2} \Gamma - \frac{1}{2} \sum_{t=2}^{T} \left[ (m_t m_t^\top + V_t) - (m_t m_{t-1}^\top + \Sigma_{t,t-1}) A^\top - A(m_t m_{t-1}^\top + \Sigma_{t,t-1})^\top + A(m_{t-1} m_{t-1}^\top + V_{t-1}) A^\top \right].
$$

(3.15)

### 3.3.2 Slicewise Conditional Likelihood Maximization

The goal of the CML learning is to find a model that minimizes the joint estimation error for the entire $y = y_1, \ldots, y_T$. However, it is often more natural to consider the prediction error at each time slice independently. In the discrete state domain, this notion is directly related to minimization of the Hamming distance between the target and the inferred states. In the continuous domain, we consider the Slicewise Conditional Likelihood Maximization (SCML) with the following objective:

$$
SCLL = \frac{1}{T} \sum_{t=1}^{T} \log P(y_t|x).
$$

(3.16)

The $SCLL$ has been introduced as an alternative objective for CRFs in the discrete domain structured classification problem [22]. Note that evaluating the objective itself requires forward/backward or filtering/smoothing. The SCML learning is also based on the gradient search.

I will extend the approach of [22] to the SSM. For notational clarity, the train data is distinguished from the random variables by denoting the former as $\bar{x}$ while the latter as $x$. It is easy to see that the $SCLL$ gradient can be written as:

$$
\frac{\partial SCLL}{\partial \theta_{ssm}} = \frac{1}{T} \sum_{t=1}^{T} \frac{\partial \log P(\bar{x}, y_t)}{\partial \theta_{ssm}} - \frac{\partial \log P(\bar{x})}{\partial \theta_{ssm}}.
$$

(3.17)

Since the second term is dealt in Eq. (3.14), I will focus on the first term of Eq. (3.17).

It can be shown that the first term excluding $(1/T)$ is equivalent to:

$$
\sum_{t=1}^{T} \int_{y \setminus y_t} P(y \setminus y_t|y_t, \bar{x}, x) \cdot \left. \frac{\partial \log P(\bar{x}, y)}{\partial \theta_{ssm}} \right|_{y_t = \bar{y}_t}.
$$

(3.18)
where $y \setminus y_t$ means set-minus, excluding $y_t$ from $y$. Recalling that the Fisher score, 
\[
\log P(x | x) \quad \text{is a sum of the 2nd-order moment terms, let } f(y_j, y_{j-1}) \text{ be one of them. This enables us to evaluate } \mathbb{E}[f(y_j, y_{j-1})] \text{ individually (with respect to the unnormalized density } \sum_{t=1}^{T} P(y \setminus y_t | y_t, x), \text{ while later on all the expectations of terms corresponding to the Fisher score have to be summed to obtain the quantity in Eq. (3.18).}
\]

For $f(y_j, y_{j-1})$, for $j = 2, \ldots, T$, the expectation $\mathbb{E}[f(y_j, y_{j-1})]$ with respect to $\sum_{t=1}^{T} P(y \setminus y_t | y_t, x)$ is:

\[
\mathbb{E}_{P(y_j | y_{j-1}, x)}[f(y_j, y_{j-1})] + \mathbb{E}_{P(y_{j-1} | y_j, x)}[f(y_j, y_{j-1})] + \sum_{t=1}^{j-2} \mathbb{E}_{P(y_j, y_{j-1} | y_t, x)}[f(y_j, y_{j-1})] + \sum_{t=j+1}^{T} \mathbb{E}_{P(y_j, y_{j-1} | y_t, x)}[f(y_j, y_{j-1})]. \tag{3.19}
\]

The first and the second terms are the expectations with respect to the posteriors given the neighbor (next or previous) state. It is not difficult to show that they are both Gaussians, namely,

\[
P(y_{t+1} | y_t, x) = \mathcal{N}(y_{t+1}; F_{t+1} y_t + b_{t+1}, R_{t+1}), \quad \text{and}
\]
\[
P(y_t | y_{t+1}, x) = \mathcal{N}(y_t; G_t y_{t+1} + c_t, S_t), \tag{3.20}
\]

where $F_{t+1} = \Sigma_{t+1} V_{t+1}^{-1}, \ G_t = \Sigma_t V_t^{-1}_{t+1}, \ b_{t+1} = m_t + 1 - F_{t+1} m_t, \ c_t = m_t - G_t m_{t+1}, \ R_{t+1} = V_{t+1} - F_{t+1} \Sigma_{t+1} V_{t+1}^{-1}, \ \text{and } S_t = V_t - G_t \Sigma_{t+1}.$

The third term of Eq. (3.19) is the expectation with respect to $P(y_j, y_{j-1} | y_t, x)$, given the state two or more slices before (note that $j > t$). This requires another forward recursion on $j$ which together with the Kalman filter forms the two-pass forward algorithm for the SCML learning. Similarly, the fourth term of Eq. (3.19) forms the second-pass backward recursion. I will derive only the forward recursion here, where the backward recursion can be similarly done. First the following lemma from the Gaussian identity is needed:

**Lemma 1.** $P(y_{t+1} | y_t, x) P(y_t | y_{t-1}, x)$ is a joint Gaussian on $(y_t, y_{t+1})$ given $(y_{t-1}, x)$. In particular, $\mu_t^1 := \mathbb{E}[y_t | y_{t-1}, x] = F_t y_{t-1} + b_t, \ mu_t^2 := \mathbb{E}[y_{t+1} | y_{t-1}, x] = F_{t+1} \mu_t^1 + b_{t+1}, \ \text{and } \forall (y_t | y_{t-1}, x) = R_t.$

Now we are ready to define the second-pass forward message as $\tilde{\alpha}_j(y_j, y_{j-1}) = \sum_{t=1}^{j-2} P(y_j, y_{j-1} | y_t, x)$, for $j = 3, \ldots, T$. It is a sum of $(j-2)$ Gaussians in the following
reason. Initially for $j = 3$, $\tilde{\alpha}_3(y_3, y_2) = P(y_3, y_2 | y_1, x)$, or equivalently, $P(y_3 | y_2, x) \cdot P(y_2 | y_1, x)$ is a Gaussian from Lemma 1. Suppose that $\tilde{\alpha}_{j-1}(y_{j-1}, y_{j-2})$ be a sum of $(j-3)$ Gaussians. In the forward recursion:

$$\tilde{\alpha}_j(y_j, y_{j-1}) = P(y_j | y_{j-1}, x) \cdot \int_{y_{j-2}} \tilde{\alpha}_{j-1}(y_{j-1}, y_{j-2}) + P(y_j | y_{j-1}, x) \cdot P(y_{j-1} | y_{j-2}, x),$$

(3.21)

the first term of RHS is a sum of $(j - 3)$ Gaussians by the inductive assumption and Lemma 1, while the second term is another Gaussian from Lemma 1. Generally, it can be shown that the $m$-th Gaussian component of $\tilde{\alpha}_j(y_j, y_{j-1})$, has the mean denoted by

$$\begin{bmatrix} \mu^1_j(m) \\ \mu^2_j(m) \end{bmatrix}$$

and the covariance by

$$\begin{bmatrix} \Sigma^{11}_j(m) & \Sigma^{12}_j(m) \\ \Sigma^{21}_j(m) & \Sigma^{22}_j(m) \end{bmatrix}$$

satisfying the recursion:

$$\begin{align*}
\mu^1_j(m) &= \mu^1_{j-1}(m), \\
\mu^2_j(m) &= F_j \mu^2_{j-1}(m) + b_j, \\
\Sigma^{22}_j(m) &= F_j \Sigma^{22}_{j-1}(m) F_j^\top + R_j, \\
\Sigma^{11}_j(m) &= \Sigma^{12}_j(m) = \Sigma^{21}_j(m) = \Sigma^{22}_{j-1}(m), \\
\Sigma^{21}_j(m) &= \Sigma^{12}_j(m)^\top = F_j \Sigma^{22}_{j-1}(m),
\end{align*}$$

(3.22)

for $m = 1, \ldots, j - 3$, and for the last ($j - 2$)-th component,

$$\begin{align*}
\mu^1_j(j - 2) &= \mu^1_{j-1}, \\
\mu^2_j(j - 2) &= \mu^2_{j-1}, \\
\Sigma^{22}_j(j - 2) &= F_j R_{j-1} F_j^\top + R_j, \\
\Sigma^{11}_j(j - 2) &= R_{j-1}, \\
\Sigma^{21}_j(j - 2) &= \Sigma^{12}_j(j - 2)^\top = F_j R_{j-1}.
\end{align*}$$

(3.23)

In the same manner, the backward message, defined as:

$$\tilde{\beta}_j(y_j, y_{j-1}) = \sum_{t=j+1}^T P(y_j, y_{j-1} | y_t, x),$$

turns out to be a sum of $(T - j)$ Gaussians. By summing up the expectations with respect to these Gaussians, Eq. (3.19) can be computed, ultimately obtaining the $SCLL$ gradient in Eq. (3.17).

Note that in the discrete case [22] has shown that the SCML learning takes the same computational complexity as the CML learning, namely the linear time in the sequence length $T$. In the continuous state domain, however, the $t$-th step in the two-pass forward/backward generates a sum of $t$ Gaussians, thereby resulting in $O(T^2)$
time. In the discrete case, the sum of $t$ multinomial probability mass functions can be represented compactly by a single multinomial mass function. In other words, the quadratic learning time originates from the difficulty in handling a sum of Gaussians at hand.

### 3.3.3 Extension to Nonlinear Dynamic Models

The CML/SCML learning can be similarly applied to the nonlinear dynamical systems (NDS). In the NDS, the posterior can be evaluated via Extended Kalman filtering/smoothing based on the approximated linear model (e.g., [13]) or using various particle filter methods, depending on the dimensionality of the state space. Since the Fisher score for NDS is no more a sum of the 2nd-order moments, rather a complex nonlinear function, evaluation of the expectation $E[f(y_t, y_{t-1})]$ is difficult. However, following [13] we can approximate arbitrary nonlinear functions by RBF networks:

$$y_t | y_{t-1} \sim \mathcal{N}(y_t; A_k k(y_{t-1}) + Ay_{t-1}, \Gamma),$$

$$x_t | y_t \sim \mathcal{N}(x_t; C_k k(y_t) + C_y, \Sigma),$$

(3.24)

where $k(y_t) := [k(y_t, u_1), \ldots, k(y_t, u_L)]^\top$ is a vector of RBF kernels evaluated on the known centers $\{u_l\}_{l=1}^L$. For $k(y_t, u_l) = e^{-\frac{1}{2}(y_t-u_l)^\top S_l^{-1}(y_t-u_l)}$, where $S_l$ is the kernel covariance, the nonlinear part in the Fisher score takes a specific form such as $k(y_t) k(y_t)^\top$, $k(y_t) k(y_{t-1})^\top$, or $y_t k(y_t)^\top$, and has a closed-form expectation with respect to a Gaussian (approximated) posterior. As a result, gradient terms necessary for the CML/SCML optimization in dynamic RBF nonlinear models also possess closed-form expressions.

In the evaluation, it is verified that for the SSM, the discriminative algorithms improve the generative learning significantly. For the NDS, however, the improvement is not as significant as the linear case. In other words, the choice of learning objective for nonlinear models appears less critical. However, the generalization performance of the nonlinear models can be very sensitive to the choice of the kernel centers and the kernel hyperparameters. In Sec. 3.5, I will demonstrate that discriminatively learned linear models can be comparable to even well-tuned nonlinear models.
3.4 Related Work

While discriminative learning of discrete-state dynamic models such as HMMs, CRFs and MEMMs has received significant attention recently, learning of similar models in the continuous space has been rarely explored. In robotics community, [1] empirically studied several objectives for learning of continuous-state dynamical systems. In contrast to [1]'s ad-hoc optimization method, it is the first to provide efficient gradient optimization algorithms for discriminative objectives, by extending the method of [22] to dynamical systems in continuous multivariate domains.

The recent work on the human motion tracking problem can be roughly categorized into: dynamic model based ([18, 39, 40]), nonlinear manifold embedding ([9, 42, 51, 66]), and Gaussian process based latent variable models ([59, 60]) to name a few. In our approach, we consider a generative family of models and show that it can be used for accurate and computationally efficient pose estimation, if coupled with a proper learning objective.

Related with the discriminative paradigm, [53] successfully employed a MEMM-like model with Bayesian mixtures of experts for 3D pose estimation. In general, MEMMs are sensitive to label-bias [27]. Their ability to successfully infer states from observations mostly depends on the modeling capacity of the regression functions and not on the choice of discriminative dynamic model objective. Unlike MEMMs, the discriminatively learned generative dynamic models could also be used for motion synthesis.

3.5 Evaluation

The discriminative dynamical system modeling approach is evaluated in a set of experiments that include synthetic data as well as the CMU motion capture dataset\textsuperscript{3}. The proposed models are denoted as CML and SCML, the SSMs learned via the methods in Sec. 3.3.1 and Sec. 3.3.2, respectively. ML is the standard maximum likelihood estimator for the SSM. I also include comparison with nonlinear and latent-variable dynamic models, as described in Sec. 3.5.2.

\textsuperscript{3}http://mocap.cs.cmu.edu/.


<table>
<thead>
<tr>
<th>Model</th>
<th>ML</th>
<th>CML</th>
<th>SCML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>1.79 ± 0.26</td>
<td>1.59 ± 0.22</td>
<td>1.30 ± 0.12</td>
</tr>
<tr>
<td>Log-Perplexity</td>
<td>4.76 ± 0.40</td>
<td>4.49 ± 0.34</td>
<td>3.80 ± 0.25</td>
</tr>
</tbody>
</table>

Table 3.1: Test errors and log-perplexities for synthetic data.

3.5.1 Synthetic Data

I synthesize data from a devised model which is structurally more complex than the SSM. The model has second-order dynamics and emission, specifically, \( y_t = \frac{1}{2}A_1 y_{t-1} + \frac{1}{2}A_2 y_{t-2} + v_t \), and \( x_t = \frac{1}{2}C_1 y_t + \frac{1}{2}C_2 y_{t-1} + w_t \), where \( v_t \) and \( w_t \) are Gaussian white noises. The purpose of this experiment is to see how the learning algorithms behave for the incorrect model structure, emphasizing the fact that it is usually difficult to figure out the correct model structure in many applications.

The evaluation is done by leave-one-out cross validation for 10 sampled sequences of lengths \( \sim N(150, 20^2) \), where \( \text{dim}(y_t) = 3 \) and \( \text{dim}(x_t) = 2 \). The test errors and the log-perplexities of three learning methods are depicted in Table 3.1. Here the estimation error is defined as an averaged norm-2 difference, \( \frac{1}{T} \sum_{t=1}^{T} \| y_t - m_t \|_2 \), where \( y_t \) is the ground truth, and \( m_t \) is the estimated state sequence. The log-perplexity is defined as \( -\frac{1}{T} \sum_{t=1}^{T} \log P(y_t | x, \theta) \). The perplexity captures the variance of the estimate, which is not characterized by the norm-2 error. The smaller number is better for both measures. The estimated sequences are also visualized in Fig. 3.2.

The result shows that the prediction performance is improved by the proposed methods, while the significance is stronger for SCML than CML. It also implies that discriminative learning can be useful for enhancing the restricted performance of the generatively trained models with (possibly) suboptimal structures.

3.5.2 Human Motion Data

I evaluate the performance of the proposed methods on the task of 3D pose estimation from real human motion data. The CMU motion capture dataset provides the ground-truth body poses (3D joint angles), which makes it possible to compare competing
methods quantitatively. Here I include three different motions: walking, picking-up a ball, and running. For each motion, 5 or 6 sequences from one subject are gathered to perform leave-one-out validation. The measurement is a 10-dim Alt-Moment feature vector extracted from the monocular silhouette image (e.g., [57]).

Typically, I will demonstrate how comparable the performance of the proposed algorithms on the SSM is to that of nonlinear models learned generatively. Two nonlinear models that are used in the evaluation are briefly discussed.

The first model is NDS described in Eq. (3.24). Since it is computationally demanding to use all poses $y_t$ in the train data for RBF kernel centers $u_l$, we instead adopt a sparse greedy kernel selection technique. It adds a pose from the pose pool (containing all train poses) one at a time, where we select the pose that maximizes a certain objective (e.g., data likelihood). Deciding the number of poses (or kernel centers) to be added is crucial for generalization performance. In the experiment, we tried for several candidates (e.g., 5%, 10%, or 20% of the pool). Then the performance of the best one for test data is reported. The kernel covariance $S_l$ for each center $u_l$ is estimated in a way that the neighbor points of $u_l$ have kernel values one half of its peak value [13]. This generates reasonably smooth kernels. Further optimization of kernel hyperparameters was not performed as it commonly results in minor performance improvements.
Table 3.2: Average test errors. The error types are abbreviated as 3 letters: The first indicates smoothed (S) or filtered (F), followed by 2 letters meaning that the error is measured in either the joint angle space (JA) or the 3D articulation point space (3P) (e.g., SJA = smoothed error in the joint angle space). The unit scale for the 3D point space is deemed as the height of the human model $\sim 25$.

<table>
<thead>
<tr>
<th>Motions</th>
<th>Err.</th>
<th>ML</th>
<th>CML</th>
<th>SCML</th>
<th>NDS</th>
<th>LVN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SJA</td>
<td>19.20</td>
<td>18.31</td>
<td>17.19</td>
<td>18.91</td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>FJA</td>
<td>22.57</td>
<td>22.73</td>
<td>20.78</td>
<td>20.84</td>
<td>19.05</td>
<td></td>
</tr>
<tr>
<td>F3P</td>
<td>20.02</td>
<td>20.28</td>
<td>17.07</td>
<td>16.59</td>
<td>14.96</td>
<td></td>
</tr>
<tr>
<td>Pick-up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SJA</td>
<td>35.03</td>
<td>33.15</td>
<td>30.56</td>
<td>33.50</td>
<td>32.23</td>
<td></td>
</tr>
<tr>
<td>FJA</td>
<td>42.28</td>
<td>38.89</td>
<td>36.99</td>
<td>41.25</td>
<td>32.10</td>
<td></td>
</tr>
<tr>
<td>S3P</td>
<td>22.60</td>
<td>21.27</td>
<td>19.33</td>
<td>21.14</td>
<td>20.49</td>
<td></td>
</tr>
<tr>
<td>F3P</td>
<td>25.20</td>
<td>24.36</td>
<td>23.83</td>
<td>25.35</td>
<td>20.40</td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SJA</td>
<td>23.35</td>
<td>22.11</td>
<td>19.39</td>
<td>21.26</td>
<td>19.08</td>
<td></td>
</tr>
<tr>
<td>FJA</td>
<td>21.87</td>
<td>22.09</td>
<td>20.92</td>
<td>21.86</td>
<td>19.76</td>
<td></td>
</tr>
<tr>
<td>S3P</td>
<td>21.52</td>
<td>19.85</td>
<td>16.96</td>
<td>18.41</td>
<td>16.97</td>
<td></td>
</tr>
<tr>
<td>F3P</td>
<td>20.40</td>
<td>20.43</td>
<td>18.43</td>
<td>18.42</td>
<td>17.65</td>
<td></td>
</tr>
</tbody>
</table>

The second model is the latent variable nonlinear dynamic model, denoted as LVN. As it is broadly believed that the realizable poses lie in a low dimensional space, it is useful to introduce latent variables $z_t$ embedded from the poses $y_t$. One possible way to devise LVN is to place dynamics on $z_t$, assuming $y_t$ and $x_t$ are generated nonlinearly (with RBF kernels) by $z_t$. Learning LVN can be done by EM algorithm on the linear approximated model as introduced in [13]. Initial subspace mapping for LVN is determined by PCA dim-reduction on the train poses. Similarly to NDS, the number of kernels is determined empirically among several candidates. In the result, we highlighted the best one.

Table 3.2 shows the average test (norm-2) errors of competing methods. We recorded the smoothed ($y_t|x$) and the filtered ($y_t|x_1,\ldots,x_t$) estimation errors for both the (joint angle) pose space and the 3D articulation point space. The latter can be easily evaluated by mapping the estimated joint angles to the body skeleton model provided in the dataset. As shown, the proposed algorithms have significantly lower prediction errors than ML learning, while exhibiting comparable (or often superior) performance to the
best nonlinear models possibly with latent variables.

It should be noticed that the filtered estimation errors of the proposed methods are not as outstanding as the smoothed ones. This is probably due to their smoothing-based objectives. It is interesting, yet left as future work, to see the performance of the modified objectives based on filtering. When comparing two discriminative algorithms, SCML yields superior performance to CML consistently for all motions. This is expected from the \textit{SCLL} objective which is more closely related with the ultimate error measure. Note also that the inference (tracking) of CML or SCML is the standard Kalman filtering/smoothing, which is much faster than the approaches based on particles or nonlinear optimization (e.g., [53, 59, 60]). In Fig. 3.3, selected frames of the estimated body skeletons are illustrated to compare SCML with the standard linear and nonlinear models.
Figure 3.3: Skeleton snapshots for walking (a–f), picking-up a ball (g–l), and running (m–s): The ground-truth is depicted by solid (cyan) lines, ML by dotted (blue), SCML by dashed (black), and latent variable nonlinear model (LVN) by dotted-dashed (red).
Chapter 4

Conditional State-Space Models

In the previous chapter, two discriminative learning algorithms, the conditional likelihood maximization (CML) and the slicewise conditional likelihood maximization (SCML), optimize $\log P(y|x, \theta)$ and $(1/T) \sum_t \log P(y_t|x, \theta)$, respectively. These approaches have shown great promise in improving prediction performance of dynamic models over those trained by the traditional generative ML learning. Unfortunately, neither of the two proposed learning objectives is convex, resulting in models that may be highly sensitive to the choice of initial parameters and the optimization procedure used for their estimation. Additionally, the models may unnecessarily, from the prediction perspective, be constrained by the generative nature of the SSM structure.

In this chapter, I propose a novel conditional undirected graphical model in the continuous state sequence domain. It relaxes generative models’ assumption on the conditional independence of the measurements given the states. To address the integrability issue, I first show that the feasible parameter space can be a convex constrained set. This enables us to formulate the parameter learning as an instance of convex optimization. In particular, I introduce a novel membership algorithm for the feasible convex set using theorems from linear algebra. Based on this membership algorithm, I provide efficient convex optimization methods for the parameter learning.

In addition, I propose a new inference algorithm for the conditional model which takes linear time in the measurement dimension as opposed to the cubic time for the Kalman filtering/smoothing for generative models. This potentially allows us to incorporate very large number of measurement features. Furthermore, I suggest a novel
discriminative feature selection algorithm which enjoys a unique merit of the regression-type undirected model like the proposed model. Finally, the performance of the proposed model is evaluated on both synthetic data and the human body pose estimation from silhouette videos.

4.1 Conditional State-Space Model (CSSM)

The proposed CSSM has an undirected graphical representation as shown in Fig. 3.1(b). Note that the model is conditioned on the entire input measurement $x$, which relaxes SSM’s assumption on the conditional independence of the measurements given the states. For discrete $y$, it resembles the CRF of [27], a well known discriminative model for the structured classification. In this sense, the CSSM can be seen as an extension of the CRF to the real-valued multivariate state domain, as is the SSM for the HMM.

For notation, input and output are vectorized, namely, $x = [x_1^T, \ldots, x_T^T]^T \in \mathbb{R}^{pT}$ and $y = [y_1^T, \ldots, y_T^T]^T \in \mathbb{R}^{dT}$. The conditional distribution of the CSSM follows a Gibbs form, $P(y|x, w) \propto e^{s(x,y;w)}$, where $s : \mathbb{R}^{pT} \times \mathbb{R}^{dT} \rightarrow \mathbb{R}$ is a score function (or a negative energy) parameterized by $w$. In particular, letting $w = \{S \in \mathbb{R}^{d \times d}, Q \in \mathbb{R}^{d \times d}, E \in \mathbb{R}^{d \times h}\}$, we define the score function as:

$$s(x,y;w) = -\frac{1}{2} \sum_{t=1}^{T} y_t^T S y_t - \sum_{t=2}^{T} y_{t-1}^T Q y_{t-1} + \sum_{t=1}^{T} y_t^T E \cdot \phi(x;t),$$

(4.1)

where $\phi(x;t)$ is a $h$-dim measurement feature vector specialized at the $t$-th position$^1$

By some algebra, we have:

$$P(y|x, \{S, Q, E\}) \propto e^{-\frac{1}{2}y^T U(S,Q)y + b(x;E)^T y},$$

(4.2)

where $U(S,Q)$ is a $(T \times T)$ block tri-diagonal matrix, and $b(x;E)$ is a $(T \times 1)$ block.

---

$^1$One can also define it to be $\phi(x)$ that depends on the entire $x$. In practice, however, one often uses a $k$-gram feature for some $k > 0$, possibly with some nonlinear mapping $\Psi$ applied to it, namely, $\phi(x;t) = \Psi(x_{t-\lfloor \frac{k}{2} \rfloor}, \ldots, x_{t+\lfloor \frac{k}{2} \rfloor}).$ I denote dim($\phi(x;t)$) by $h$. 

vector as follows:

\[
U(S, Q) = \begin{bmatrix}
S & Q^\top & 0 & \cdots \\
0 & Q & S & Q^\top & \cdots \\
& 0 & Q & S & \cdots \\
& & & \ddots & \ddots & \ddots \\
& & & & \ddots & \ddots & \ddots \\
\end{bmatrix}, \quad b(x; E) = \begin{bmatrix}
E \cdot \phi(x; 1) \\
\vdots \\
E \cdot \phi(x; T)
\end{bmatrix}.
\] (4.3)

Note that defining the score function as in Eq. (4.1) corresponds to have linear Gaussian dynamics features, \( \varphi(y_t, y_{t-1}) := \{y_t, y_t^\top, y_t y_{t-1}^\top\} \), and (non)-linear emission features, \( \varphi(y_t, x) := y_t \cdot \phi(x; t)^\top \). Even though one can define arbitrary triangle features, \( \varphi(y_t, y_{t-1}, x) \), in order to fully respect the graph structure in Fig. 3.1(b), there are obvious advantages of this choice of the features.

First, the score is a quadratic function in terms of \( y \), which implies that \( P(y|x) \) in Eq. (4.2) is a Gaussian provided that the second-order coefficient matrix \( U(S, Q) \) is positive (semi)-definite. This enables us to enjoy many nice properties of Gaussian. Most of all, we guarantee that the distribution is proper, meaning that it is integrable (i.e., the partition function, \( \int y e^{s(x,y)} \), is finite). Moreover, the decoding of the states, namely, \( \arg\max_y s(x,y) \), is unique to the global optimum, and can be done efficiently (in \( O(T) \)) by the inference algorithm introduced in Sec. 4.1.3.

Second, by defining the dynamics features solely based on \( y \), we have \( U(S, Q) \) independent on \( x \), which makes it possible to pose \( U(S, Q) \succeq 0 \) regardless of \( x \). In other words, the feasible parameter space, \( \{(S, Q, E)|U(S, Q) \succeq 0\} \), is not affected by \( x \), allowing that a trained model can be used for unseen test measurement without worrying about the integrability. How the constraint \( U(S, Q) \succeq 0 \) can be posed efficiently (in terms of \( S \) and \( Q \)) will be discussed in Sec. 4.1.2.

In fact, this choice of the features is reasonable in the following sense: When ignoring the emission features \( \varphi(y_t, x) \), the model is roughly\(^2\) equivalent to the AR model. On the other hand, by dropping the dynamics features \( \varphi(y_t, y_{t-1}) \), and further assuming that \( \phi(x; t) = [\ldots, k(x_t, x_j), \ldots]^\top \) for some kernel function \( k(\cdot, \cdot) \), the model reduces to a multivariate kernel regression model with i.i.d. pairs \( \{(y_t, x_t)\}_{t=1}^T \) across \( t \).

\(^2\)There exists some discrepancy at the boundaries \( t = 1 \) and \( t = T \).
4.1.1 Feasible Parameter Space

I first specify the feasible parameter space in which the parameter learning will take place. The feasible parameters are those that make the Gaussian in Eq. (4.2) have a positive definite precision matrix \( U_T(S,Q) \). Here we use the subscript \( T \) to indicate a \((T \times T)\) block matrix for the length-\( T \) sequence. As we do not know a priori the test sequence length, it is reasonable to define the feasible parameter space as

\[
W = \{(S, Q, E) | U_T(S, Q) \succeq 0, \forall T > 0\}.
\]

Since we have an inclusion relation that \( U_T \succeq 0 \) implies \( U_{T+1} \succeq 0 \), it is easy to see that \( W = \{(S, Q, E) | U_\infty(S, Q) \succeq 0\} \). By abusing notation, we use \( U \) to indicate \( U_T \) for either a fixed \( T \) or arbitrary large \( T \), which is clear in the context.

4.1.2 Parameter Learning

We consider learning the parameters \( w = (S, Q, E) \) of the model for a given train data \( \{(x^i, y^i)\}_{i=1}^n \). One may expect to exploit the Gaussian properties in learning as

\[
P(y|x, w) = \mathcal{N}(y; U(S, Q)^{-1} \cdot b(x; E), U(S, Q)^{-1}).
\]

However, this is precluded by the following reasons: (1) the dimensions of data (or the sizes of \( U(S, Q) \)) are not the same due to potentially unequal length sequences, (2) inverting \( U(S, Q) \) is usually infeasible, and (3) constraining the structure of \( U(S, Q) \) (Eq. (4.3)) is difficult.

Instead, we optimize the log-likelihood\(^4\) with respect to \( w \in W \):

\[
\min_w -\sum_{i=1}^n \log P(y^i|x^i, w), \quad \text{s.t. } w \in W. \tag{4.4}
\]

Note that the negative log-likelihood is convex in \( w \) as it can be decomposed into the linear negative score function and the convex log-partition function as follows:

\[
-\log P(y|x, w) = -s(x, y; w) + \log Z(x; w),
\]

where the partition function \( Z(x; w) = \int_y e^{s(x, y; w)} \). Furthermore, it is easy to see that the feasible constraint set \( W \) is a convex (cone) in terms of \( w \). To see this, for

\(^3\)It is because \( U_T \) is a principal sub-matrix of \( U_{T+1} \).

\(^4\)Perhaps one may add a penalty term \( \lambda \|w\|^2 \) for some scale hyperparameter \( \lambda \), however, I skip it in the derivation for simplicity. Note that adding the term does not affect the convexity of the cost function.
\( w_1 = (S_1, Q_1, E_1) \in \mathcal{W} \) and \( w_2 = (S_2, Q_2, E_2) \in \mathcal{W} \), note that \((\nu \cdot w_1 + \eta \cdot w_2) \in \mathcal{W}\) for any \( \nu, \eta \geq 0 \). Thus Eq. (4.4) is an instance of convex optimization: convex cost + convex conic constraint.

The major concern in this optimization is how to pose the constraint \( U(S,Q) \succeq 0 \) efficiently. The straightforward approach is to consider \( U \) as an arbitrary positive definite matrix with a set of linear constraints to define the structure of \( U \) in Eq. (4.3).

More specifically,

\[
\begin{align*}
U_{I,I+1} &= U_{I-1,I}, \quad U_{I+1,I} = U_{I,I-1}, \quad \text{for } I \geq 2, \\
U_{1,2} &= U_{2,1}^\top, \quad U_{I,J} = 0_{d \times d}, \quad \text{for } |I - J| \geq 2, \\
U_{I,J} &= U_{I-1,J-1}, \quad \text{for } I \geq 2, \quad \text{and } U \succeq 0,
\end{align*}
\]

where \( U_{I,J} \) is the \((I, J)\)-th block of \( U \). However, plugging it into the general convex solvers would be too slow. Moreover, the popular method like back-projection by eigenvalue truncation (e.g. [71]) for the positive definite constraint may be intractable due to the size of \( U \).

Rather, I propose a novel optimization method based on the gradient descent, where the gradient is taken with respect to \( w = (S, Q, E) \), instead of the whole block matrix \( U \). The gradient of Eq. (4.5) is denoted as \( \nabla_w = \{\partial S, \partial Q, \partial E\} \), where

\[
\begin{align*}
\partial S &= \frac{1}{2} \sum_{t=1}^T \left( y_t y_t^\top - \mathbb{E}[y_t y_t^\top | x] \right), \\
\partial Q &= \sum_{t=2}^T \left( y_t y_{t-1}^\top - \mathbb{E}[y_t y_{t-1}^\top | x] \right), \\
\partial E &= - \sum_{t=1}^T \left( y_t \cdot \phi(x; t)^\top - \mathbb{E}[y_t | x] \cdot \phi(x; t)^\top \right).
\end{align*}
\]

The posterior means that appear in \( \nabla_w \) can be easily computed by the inference algorithm in Sec. 4.1.3. Following the general gradient descent approach, the next iterate is determined as \( w_\eta^{\text{new}} = w + \eta \cdot G(\nabla_w) \), where \( \eta > 0 \) is a chosen step-size, and \( G(\nabla_w) \) is an operator that determines the descent direction. For instance, \( G(\nabla_w) = -\nabla_w \) for the steepest descent while \( G(\nabla_w) = -H_w^{-1} \cdot \nabla_w \) for the Newton method, where \( H_w \) is the Hessian at \( w \).
Algorithm 1: Membership to $\mathcal{W}$

Input: $S$ and $Q$.
Output: TRUE if $U(S,Q) \succeq 0$, FALSE otherwise.
if $S \not\succeq 0$ then
    return FALSE.
end if
$B = S^{-1/2}QS^{-1/2}$.
if $\rho(B) > 1/2$ then
    return FALSE.
end if
if $\|B\|_2 \leq 1/2$ then
    return TRUE.
end if
$L_0 = I$. Choose a sufficiently large number $J$.
for $j = 1, \ldots, J$ do
    $L_j = I - B^\top L_{j-1} B$.
    if $L_j \not\succeq BB^\top$ then
        return FALSE.
    end if
end for
return TRUE.

The main idea is to adjust the step size $\eta$ so that $w_\eta^{\text{new}}$ belongs to $\mathcal{W}$. For some trial $\eta$, if $w_\eta^{\text{new}} \not\in \mathcal{W}$, then we reduce $\eta$ (e.g., $\eta \leftarrow \frac{1}{2} \eta$). The reason we do not enlarge $\eta$ is that due to the convexity of $\mathcal{W}$, $w_\nu^{\text{new}} \not\in \mathcal{W}$ for any $\nu \geq \eta$, otherwise $w_\eta^{\text{new}}$ that lies on the line segment ending at $w$ and $w_\nu^{\text{new}}$ should have resided in $\mathcal{W}$. If $w_\eta^{\text{new}} \in \mathcal{W}$, we do a line-search: find $\eta^* \in [0, \eta]$ that minimizes the cost. Note that any step size in $[0, \eta]$ results in a feasible model due to convex $\mathcal{W}$. See Fig. 4.1.

This approach requires an efficient membership algorithm that determines whether $U(S,Q) \succeq 0$ or not, for arbitrary $S$ and $Q$. Fortunately, from linear algebra there are some known theorems that study $U$ which is also known as a special type of block Toeplitz matrix. The following theorem is from [10, 11] where its proof is therein.

Theorem 2. $U(S,Q) \succeq 0$ if and only if $L_j \succeq BB^\top$ for all $j \geq 1$, where $B = S^{-1/2}QS^{-1/2}$, and $\{L_j\}$ is recursively defined as: $L_0 = I$, $L_j = I - B^\top L_{j-1} B$, $j \geq 1$. ($I$ is the $(d \times d)$ identity matrix.)

Notice that checking the condition $L_j \succeq BB^\top$ for $(d \times d)$ matrices is quite easy. The following theorem provides a sufficient condition and a necessary condition which
are both non-iterative (See [10] for the proofs).

**Theorem 3.** *(Sufficiency)* \( \mathbf{U}(\mathbf{S}, \mathbf{Q}) \succeq 0 \) if \( \|\mathbf{B}\|_2 \leq 1/2 \). *(Necessity)* \( \rho(\mathbf{B}) \leq 1/2 \) if \( \mathbf{U}(\mathbf{S}, \mathbf{Q}) \succeq 0 \).

Here, \( \|\mathbf{B}\|_2 \) is the 2-norm (or the largest singular value of \( \mathbf{B} \)) and \( \rho(\mathbf{B}) \) is the spectral radius of \( \mathbf{B} \) (or the largest eigenvalue magnitude).

The iterative membership algorithm (See Alg. 1) is based on Thm. 2, while the conditions in Thm. 3 are also used to quickly filter out the members or the non-members for \( \mathcal{W} \). In our experiments, the maximum number of iterations chosen as \( J = 1,000 \) works well without feasibility violation. The overall learning algorithm that uses the membership algorithm with a bisection step-size adjustment is described in Alg 2.

**Algorithm 2** CSSM Learning via Step-Size Adjustment

(a) Initialize \( \mathbf{w} \).
(b) Compute \( \nabla \mathbf{w} \).
(c) \( \eta \leftarrow 1 \).
(d) \( \mathbf{w}_\eta = (\mathbf{S}_\eta, \mathbf{Q}_\eta, \mathbf{E}_\eta) \leftarrow \mathbf{w} + \eta \cdot \mathcal{G}(\nabla \mathbf{w}) \).

if Membership\((\mathbf{S}_\eta, \mathbf{Q}_\eta)\) == TRUE then
   Do line-search to find \( \eta^* \).
   \( \mathbf{w} \leftarrow \mathbf{w} + \eta^* \cdot \mathcal{G}(\nabla \mathbf{w}) \). goto (b).
else
   \( \eta \leftarrow \eta/2 \). goto (d).
end if
4.1.3 Inference and Decoding

The inference is the task to compute the posterior distribution, \( P(y|x) \), for the given \( x \), while the decoding is to find its mode, \( y^* = \arg \max_y P(y|x) \). In the CSSM, due to the conditional Gaussianity (Eq. (4.2)) and the chain structure (Fig. 3.1(b)), the Gaussian posteriors on the cliques, namely, \( P(y_t|x) \) and \( P(y_t,y_{t-1}|x) \) completely represents the full posterior \( P(y|x) \). Moreover, the decoding coincides with the posterior means, that is, \( y^*_t = \mathbb{E}[y_t|x] \). The well-known Kalman filtering/smoothing is the algorithm to compute \( P(y_t|x) \) and \( P(y_t,y_{t-1}|x) \). Here we derive an alternative approach based on the general message passing for the undirected graphical model.

For the given \( x \), the potential function \( M_t(\cdot) \) defined on the clique at time \( t \) is:

\[
M_t(y_t,y_{t-1}) = e^{-\frac{1}{2}y_t^T S y_t - y_{t-1}^T Q y_{t-1} + y_t^T E \phi(x;t)}, \quad t \geq 2,
\]

\[
M_1(y_1) = e^{-\frac{1}{2}y_1^T S y_1 + E \phi(x;1)}.
\]  

(4.8)

The forward messages are recursively defined with the initial condition, \( \alpha_1(y_1) = M_1(y_1) \), and for \( t = 2, \ldots, T \),

\[
\alpha_t(y_t) = \int_{y_{t-1}} \alpha_{t-1}(y_{t-1}) \cdot M_t(y_t,y_{t-1}).
\]  

(4.9)

Since \( \alpha_t(y_t) \) is an unnormalized Gaussian, it can be represented by a triplet, \( (r_t, P_t, q_t) \) \( \in \mathbb{R} \times \mathbb{R}^{d 	imes d} \times \mathbb{R}^d \), which implies \( \alpha_t(y_t) = r_t \exp(-\frac{1}{2}y_t^T P_t y_t + q_t^T y_t) \). Following the recursion in Eq. (4.9), we can compute \( (r_t, P_t, q_t) \) recursively as follows:

\[
t = 1; \quad r_1 = 1, \quad P_1 = S_1, \quad q_1 = E \cdot \phi(x;1),
\]

\[
t \geq 2; \quad r_t = r_{t-1} \cdot |2\pi P_{t-1}^{-1}|^{1/2} \cdot e^{\frac{1}{2}y_t^T P_{t-1}^{-1} y_t} \cdot e^{\frac{1}{2}q_t^T q_{t-1} P_{t-1}^{-1} q_{t-1}},
\]

\[
P_t = S - Q P_{t-1}^{-1} Q^T,
\]

\[
q_t = E \cdot \phi(x;t) - Q P_{t-1}^{-1} q_{t-1}.
\]  

(4.10)

It is important to notice that Eq. (4.10) makes sense only if the integrability condition in Eq. (4.9), that is, \( P_t \succeq 0 \) for all \( t \), is satisfied. we can guarantee that for the feasible model that resides in \( \mathcal{W} \), this condition always holds. The partition function is
easily obtained from the forward messages, in particular, $Z(x) = \int_{y_T} \alpha_T(y_T)$ .

Similarly, the backward messages can be defined with the initial, $\beta_T(y_T) = 1$, and for $t < T$,

$$\beta_t(y_t) = \int_{y_{t+1}} \beta_{t+1}(y_{t+1}) \cdot M_{t+1}(y_{t+1}, y_t).$$  \hfill (4.11)

The triplet, $(h_t, F_t, g_t) \in (\mathbb{R}, \mathbb{R}^{d \times d}, \mathbb{R}^d)$, for representing $\beta_t(y_t) = h_t \exp(-\frac{1}{2} y_t^\top F_t y_t + g_t^\top y_t)$, can be computed as:

$$t = T; \quad h_T = 1, \quad F_T = 0_{d \times d}, \quad g_T = 0_{d \times 1},$$

$$t < T; \quad h_t = h_{t+1} \cdot \sqrt{2\pi F_{t+1}^{-1}} \cdot e^{\frac{1}{2} g_{t+1}^\top F_{t+1}^{-1} g_{t+1}},$$

$$F_t = -Q^\top F_{t+1}^{-1} Q, \quad g_t = -Q^\top F_{t+1}^{-1} g_{t+1},$$  \hfill (4.12)

where $\tilde{F}_t := F_t + S$ and $\tilde{g}_t := g_t + E \cdot \phi(x; t)$. Another integrability condition in Eq. (4.11), $\tilde{F}_t \succeq 0$ for all $t$, also holds for the feasible parameters.

Observing that $P(y_t|x) = \frac{1}{Z(x)} \alpha_t(y_t) \cdot \beta_t(y_t)$ and $P(y_t, y_{t-1}|x) = \frac{1}{Z(x)} \alpha_{t-1}(y_{t-1}) \cdot M_t(y_t, y_{t-1}) \cdot \beta_t(y_t)$, we have the following Gaussian posterior:

$$P(y_t, y_{t-1}|x) = \mathcal{N}\left(\begin{bmatrix} m_t \\ m_{t-1} \end{bmatrix}, \begin{bmatrix} V_t & V_{t,t-1} \\ V_{t,t-1}^\top & V_{t-1} \end{bmatrix}\right),$$  \hfill (4.13)

where $m_t = \mathbb{E}[y_t|x] = (P_t + F_t)^{-1} \cdot (q_t + g_t)$, $V_t = \mathbb{V}(y_t|x) = (P_t + F_t)^{-1}$, and $V_{t,t-1} = \text{Cov}(y_t, y_{t-1}|x) = -\tilde{F}_t^{-1} Q (P_{t-1} - Q^\top \tilde{F}_t^{-1} Q)^{-1}$. Notice that for a feasible model, all the posterior quantities are proper (i.e., covariances are positive definite) since they are marginals of the full Gaussian (Eq. (4.2)).

It is interesting to consider the proposed inference algorithm in light of the standard Kalman smoothing for the SSM. In time complexity, the proposed method takes $O(T \cdot (d^3 + dh))$ time, while the Kalman smoothing requires $O(T \cdot (d^3 + d^3 + dh))$ , where we assume that the corresponding SSM employs $\phi(x; t)$ as its $t$-th observation. The linear time in the measurement dimension enables the CSSM to incorporate a large number of measurement features. Empirically, we observe that the proposed inference

---

5 This must be the same to the Gaussian normalizing constant from Eq. (4.2), namely, $(2\pi)^{dT/2} \cdot |U|^{-1/2} \cdot \exp(\frac{1}{2} \mathbf{b}^\top U^{-1} \mathbf{b})$.

6 Recently, it has been shown that the cubic in $h$ can be reduced to quadratic, using the least-square method. See [62] for details.
algorithm in conjunction with the step size adjustment (Sec. 4.1.2) is numerically more stable than the Kalman smoothing for SSM, especially when $d$ is large.

4.2 Variants of CSSM

In this section I discuss two extensions of the CSSM: the feature selection and inclusion of the nonlinear dynamics features.

4.2.1 Discriminative Feature Selection

A typical benefit of regression-type undirected conditional models is their eligibility for feature selection, especially when the data contains a combination of relevant and a large number irrelevant (noisy) features. Given a pool of candidate features denoted as $\mathcal{F}$, the discriminative feature selection algorithm selects a sparse subset that contains the most salient and discriminative features.

The algorithm is essentially a greedy forward selection (boosting): at each stage, a new feature is added to the current feature set. We follow [28]'s functional gradient derivation to compute the gain for each candidate feature $f \in \mathcal{F}$. For the current feature set $\phi(x)$, we have the following criterion (gain) for $f$:

$$L(f; \phi(x)) = \left\| \sum_t f(x; t) \cdot (y_t - \mathbb{E}_{\phi(x)}[y_t|x]) \right\|_1, \quad (4.14)$$

where the expectation is taken with respect to the learned model with the current feature set $\phi(x)$.

The gain can be interpreted as a discrepancy between the empirical features and the model expected features. However, Eq. (4.14) is slightly different from the derivation in [28] in that we take $\| \cdot \|_1$ to make the multivariate quantity to scalar. Notice that if the state is discrete as in [28], the quantity in $\| \cdot \|_1$ is scalar. Once the gains are computed, the new feature to be added is $f^* = \max_{f \in \mathcal{F}} L(f; \phi(x))$.

This also makes sense intuitively because the feature that is the most positively correlated with the current error is the one to be selected. However, according to Eq. (4.14), simply scaling up $f$ results in a better feature. It is thus crucial to normalize the features before computing the gains, namely, $f(x; t) = f(x; t)/\sum_t f(x; t)$. 
After adding the feature \( f^* \) to the model, we exclude it from \( \mathcal{F} \), followed by learning parameters of the new model.

### 4.2.2 Nonlinear Dynamics

Although the CSSM is derived for linear dynamics features, it can be easily extended to incorporate nonlinear dynamics features. Motivated by [52], one can use an invertible nonlinear mapping \( \chi(\cdot) \) to obtain a new embedded state \( \zeta_t = \chi(y_t) \). For instance, if the mapping is related to a polynomial kernel with an odd degree, it is invertible and the inversion can be found in a closed form. In the embedded space we may assume that \( \zeta \) conforms to linear dynamics despite nonlinear dynamics in the original space. In the linear \( \zeta \) space, all the previous derivations do not change. It is only required to transform the estimated \( \zeta \) back to the original space: \( y_t = \chi^{-1}(\zeta_t) \).

### 4.3 Related Work

While discriminative modeling of discrete-state dynamics such as CRFs has received significant attention recently, similar models in the continuous state space has been rarely explored. Recent works that deal with discriminative dynamics models in continuous domain do not consider the integrability (or feasibility) conditions on the parameter space discussed here. Hence such approaches had to resort to sampling or discretization to circumvent the integrability issue. For instance, [44] introduced a novel conditional model with switching latent variables, however, the inference is done by MCMC which would be computationally intensive for high dimensions. [55] tried to discretize the continuous state space into grids, which may require a huge number of poses to be known in order to have a good approximation.

In robotics community, [1] empirically studied several objectives for learning of dynamical systems. In contrast to [1]'s ad-hoc optimization method, [24] (also described in the previous chapter) provided efficient gradient-based optimization algorithms for discriminative objectives to dynamical systems. Compared to the CSSM, they are
optimized with respect to the parameters of the generative model, which results in non-convex optimization. [53] successfully extended MEMM of [35] with Bayesian mixtures of experts for 3D pose estimation. However, their ability to successfully infer states from observations mostly depends on the modeling capacity of the regression functions, not on the choice of the discriminative dynamic model objective.

4.4 Evaluation

The performance of the proposed models is evaluated on synthetic data, robot-arm state prediction, and human motion estimation tasks. I focus on comparison between the generative SSMs trained by either the standard ML learning or the discriminative learning (CML and SCML), and the proposed CSSMs with the convex CML learning and the SCML learning. I use a “model-learning” notation to denote a specific method (e.g., “SSM-CML” means SSM learned by CML). In all experiments, iterative estimators are initialized with SSM-ML parameters. Furthermore, the learned SSMs and CSSMs are assumed to have 1st-order linear dynamics, a possible mismatch with the true data dynamics.

4.4.1 Synthetic Data

The synthetic data is generated from the dynamic model described in Sec. 3.5.1. Although the SSM-CML and the CSSM-CML optimize the same objective, the SSM-CML learning is in general non-convex with many local optima. When the learning starts from the SSM-ML parameters, both models exhibit very similar test errors as shown in Table 4.1. Then to see the difference, both models are learned with the same initial parameters that are randomly chosen. The CSSM-CML recorded test error 1.83 which is not very close to the one at the global optimum, however, this is much better than the test error 2.30 of the SSM-CML. This implies that the CSSM is less sensitive to the initial iterate.

As the result in Table 4.1 demonstrates, SCML is empirically more robust than the

---

7It is straightforward to find the CSSM that corresponds to the SSM-ML parameters.
<table>
<thead>
<tr>
<th>Model</th>
<th>CSSM-CML</th>
<th>SSM-ML</th>
<th>SSM-CML</th>
<th>SSM-SCML</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2-Error</td>
<td>1.54 ± 0.21</td>
<td>1.79 ± 0.26</td>
<td>1.59 ± 0.22</td>
<td>1.30 ± 0.12</td>
</tr>
<tr>
<td>Log-Perp.</td>
<td>4.26 ± 0.35</td>
<td>4.76 ± 0.40</td>
<td>4.49 ± 0.34</td>
<td>3.80 ± 0.25</td>
</tr>
</tbody>
</table>

Table 4.1: Test errors and log-perplexities on synthetic data.

CML and the CSSM, especially with sub-optimal model structures [22, 24]. However, its learning time is quadratic in sequence length, which may prevent it from being applied to large scale data.

**Feature Selection**

Next I apply the feature selection algorithm discussed in 4.2.1. In addition to the original \((p = 2)\)-dim measurement features, I add extra 20 irrelevant noisy features that are generated randomly. First, the SSM is learned via ML, CML, and SCML with this 22-dim feature set, where the test L2 errors are 2.06, 1.92, and 1.67, respectively. On the other hand, the CSSM feature selection algorithm (starting from empty feature) successfully finds the two original (relevant) features for its first two stages. Fig. 4.2 shows the features found by our feature selection algorithm and the corresponding test errors.

**4.4.2 Robot-Arm Data**

The dataset is a modification of the 2D nonlinear robot arm mapping problem defined in [34]. The original data was constructed for static regression tests, where the arm-end effector \(\mathbf{x}\) is determined by two joint angles \(y_1\) and \(y_2\), via the following equation:

\[
\mathbf{x} = r_1 \cos(y_1) + r_2 \cos(y_1 + y_2) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2).
\]

(4.15)

I modify the problem by augmenting the original i.i.d. sampling mechanism for \(\mathbf{y} = [y_1, y_2]^\top\) (denoted by (a) in the subsequent discussion) with (b) 1st-order or (c) 2nd-order temporal dynamics. In particular, the transition matrix of (b) is a slightly perturbed orthonormal matrix that rotates \(\mathbf{y}_{t-1}\) around a fixed center to yield \(\mathbf{y}_t\). In (c)
Figure 4.2: Feature selection. The solid (cyan) line indicates test L2 errors of the CSSM learned with the features selected by the proposed algorithm. It starts from empty feature, adds a feature at each stage, up to full features (22nd stage). Each number indicates the feature index, where the original features correspond to 1 and 2. The test errors of the SSM learned with full 22 features are depicted: SSM-ML by dotted (blue), SSM-CML by dotted-dashed (red), and SSM-SCML by dashed (black).

$y_t$ is affected by both the rotation of $y_{t-2}$ and a non-orthonormal transformation of $y_{t-1}$. See Fig. 4.3 for the generated samples.

Seven sequences of length $\sim 200$ were sampled in the three settings. For a baseline comparison, I also test the static least-square linear/kernel regression models which regard the slices of a sequences as i.i.d. samples. Table 4.2 shows the performance of the competing models, where the error measure used is the norm-2 difference averaged over time slices$^8$.

In (a), the performance of all methods is indistinguishable. Because the states $y_t$ are independent, the dynamic models show no benefit over the static regression. In (b), where the dynamics in data exactly match that of the models, all SSMs and CSSMs work equally well, yielding significant improvement over the static models. However, in (c), the case of suboptimal model dynamics (1st-order model structure vs. 2nd-order generating process) the SSM-ML performs poorly as it attempts to fit the data jointly to the incorrect model structure. In contrast, the discriminative learning of conditional

$^8(1/T)\sum_{t=1}^{T}||\mathbf{y}_t - \hat{\mathbf{y}}_t||_2$, where $\mathbf{y}$ is the ground truth, and $\hat{\mathbf{y}}$ is the estimated state.
Figure 4.3: Robot-Arm data samples.

Table 4.2: Test L2 errors for robot-arm data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SSM-ML</th>
<th>SSM-CML</th>
<th>SSM-SCML</th>
<th>CSSM-CML</th>
<th>CSSM-SCML</th>
<th>Linear Regression</th>
<th>Kernel Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) I.I.D.</td>
<td>0.6124 ± 0.0141</td>
<td>0.6121 ± 0.0140</td>
<td>0.6120 ± 0.0152</td>
<td>0.6116 ± 0.0158</td>
<td>0.6121 ± 0.0156</td>
<td>0.6133 ± 0.0124</td>
<td>0.5994 ± 0.0267</td>
</tr>
<tr>
<td>(b) 1st-Order</td>
<td>0.0590 ± 0.0042</td>
<td>0.0698 ± 0.0099</td>
<td>0.0567 ± 0.0028</td>
<td>0.0571 ± 0.0051</td>
<td>0.0542 ± 0.0030</td>
<td>0.3169 ± 0.0100</td>
<td>0.2887 ± 0.0086</td>
</tr>
<tr>
<td>(c) 2nd-Order</td>
<td>0.8916 ± 0.0051</td>
<td>0.5033 ± 0.0054</td>
<td>0.3942 ± 0.0102</td>
<td>0.5358 ± 0.0104</td>
<td>0.4515 ± 0.0070</td>
<td>1.1317 ± 0.0040</td>
<td>0.8783 ± 0.0080</td>
</tr>
</tbody>
</table>

models is viable focusing on the prediction task more effectively. Interestingly, the non-convex SSM-CML is slightly better than CSSM-CML, which may be possible under the wrong model assumption.

The result also demonstrates that the SCML learning (for both SSMs and CSSMs) yields lower errors than the CML (more significantly in (c)), which implies that the SCML is more robust to sub-optimal model structures. However, one should consider the trade-off between robustness and computational overhead of the SCML learning. Compared to (a), accuracy of the static regression and the SSM-ML degrades. This may be expected because of the strong mismatch between the state generating processes and the model structure.
Figure 4.4: Feature selection with the RBF extended feature set. The left is the dataset (b) and the right is the dataset (c). See text for details.

Feature selection

Then the feature selection algorithm of 4.2.1 is applied. I focus on two scenarios where the selection is typically advantageous: incorporating a rich set of measurements and eliminating less salient features. In the first setting I form a pool of $L + 1$ features: the original measurement $x_t$ and the nonlinear RBF features $\{k(x_t, u_l)\}_{l=1}^L$ at time $t$. The $L$ center positions $\{u_l\}_{l=1}^L$ are obtained by clustering the train data $x$. I choose $L = 50$, indexing the original feature $x$ as 1.

Fig. 4.4 shows errors of the forward feature selection on datasets (b) and (c). The numbers indicate the feature indices added at each stage (every third one is shown). The dashed line depicts the test error of the best method without RBF features (from Table 4.2). One can see that the original feature (id=1) is always discovered in the first stage. Moreover, incorporating nonlinear features improves the prediction performance significantly. More importantly, the feature selection finds a sparse set of the most salient features without severe penalty in test accuracy. E.g., 30% of features from the pool in (c) yields the same performance as the model with the full set. The reduced set also results in significant computational savings in the inference phase.

In the second setting, I demonstrate the benefit of feature selection in terms of eliminating irrelevant features to prediction. I generate 9 noisy dimensions stacked to
<table>
<thead>
<tr>
<th>CSSM-CML</th>
<th>SSM-ML</th>
<th>SSM-CML</th>
<th>SSM-SCML</th>
<th>NDS</th>
<th>LVN</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.85</td>
<td>19.20</td>
<td>18.31</td>
<td>17.19</td>
<td>18.91</td>
<td>18.01</td>
</tr>
</tbody>
</table>

Table 4.3: Average test L2 errors for CMU walking motion data.

the original $x$. The noise is stationary white or non-stationary impulse noise\(^9\). The use of all 10 features results in considerable degrade in test accuracy. For instance, the CSSM-CML recorded test errors $0.1090 \pm 0.0392$ (stationary) and $0.0818 \pm 0.0166$ (non-stationary noise) for dataset (b). On the other hand, the feature selection algorithm successfully finds the original feature at the first stage, eliminating the adverse noise effects.

### 4.4.3 Human Motion Data

I evaluate the performance of the proposed discriminative model on the task of 3D body pose estimation from human motion data. For the walking motion, the same setting as in Chap. 3 is used. Table 4.3 (directly comparable to the first row of Table 3.2) shows the average test L2 errors of the competing methods. The CSSM-CML has significantly lower errors than the SSM-ML and more importantly, the SSM-CML. This indicates that the discriminative model can achieve significant improvement beyond the discriminative learning of generative models. The CSSM also has superior performance to the nonlinear models. This implies that the discriminative model even with a simple linear structure can have potential to outperform complex nonlinear models in terms of generalization performance.

\(^9\)Modeled by a 2-state Markov process: one state emits a small amount of noise more frequently than the other that generates high noise bursts.
Chapter 5

Dimensionality Reduction for Regression

The task of *dimensionality reduction for regression* (DRR) is to find a low dimensional representation, $z \in \mathbb{R}^q$, of the input covariates, $x \in \mathbb{R}^p$, with $q \ll p$, for regressing the output, $y \in \mathbb{R}^d$, given $n$ i.i.d. data $\{(x^i, y^i)\}_{i=1}^n$. DRR has found many applications in visualization of high dimensional data, efficient regressor design with a reduced input dimension, and elimination of noise in data $x$ by uncovering the essential information $z$ for predicting $y$. In all these tasks DRR is not tied to a particular regression estimation method, but can be rather seen as a prior task to the regressor design for a better understanding of data.

DRR differs from other well-known dimensionality reduction algorithms in several ways. One can view DRR as a *supervised* learning technique with *real multivariate* labels $y$. Most other supervised techniques focus on the classification setting (i.e., discrete $y$), including Linear Discriminant Analysis (LDA), kernel LDA, general graph embedding [72], and metric learning [14, 67, 71]. Unsupervised dimension reduction methods, on the other hand, assume that $y$ is unknown. Principal subspace methods (PCA and kernel PCA [48]), nonlinear locality-preserving manifold learning (LLE [45], ISOMAP [56], and Laplacian Eigenmap [4]), and probabilistic methods like GPLVM [30] belong to this class of approaches that do not leverage known target values. DRR has been a focus of several important lines of research in the statistical machine learning community ([5, 12, 33, 38]). However, it has received significantly less attention in the domain of computer vision.

The crucial notion related to DRR is that of *sufficiency in dimension reduction* (SDR, [5, 12, 33]). SDR states that one has to find the linear subspace bases $B = [b_1, \ldots, b_q]$ with $b_l \in \mathbb{R}^p$, (or basis functions in the nonlinear case, $B = \{b_1(\cdot), \ldots, b_q(\cdot)\}$)
such that $y$ and $x$ are conditionally independent given $B^T x$. As this condition implies that the conditional distribution of $y$ given $x$ equals that of $y$ given $z = B^T x$, the dimension reduction entails no loss of information for the purpose of regression. It is known that such $B$ always exists (at least the identity $B = I$ for $q = p$) with non-unique solutions\(^1\). Hence, one is naturally interested in the minimal subspace or the intersection of all such subspaces, often called the *central subspace* \(^2\).

Two schools of approaches have been suggested to find the central subspace: the inverse regression (IR) \([33, 70]\) and the kernel dimension reduction (KDR) \([12, 38]\). KDR \([12]\) directly reduces the task of imposing conditional independence to the optimization problem that minimizes the conditional covariance operator\(^3\) in a RKHS (reproducing kernel Hilbert space). This is achieved by quantifying the notion of conditional dependency (between $y$ and $x$ given $B^T x$) using a positive definite ordering of the expected covariance operators in what is called the probability-determining RKHS (e.g., the RBF kernel-induced Hilbert space).

Although KDR formulates the problem in RKHS, the final projection is linear in the original space. For a nonlinear extension, \([38]\) proposed the manifold KDR which first maps the original input space to a nonlinear manifold (e.g., by Laplacian Eigenmap learned from $x$ only), and applies the KDR to find a linear subspace in the manifold. However, this introduces a tight coupling between the central subspace and the separately learned input manifold, restricting the approach to a transduction setting. That is, for a new input point, one has to rebuild the manifold entirely with data including the new point\(^4\). Moreover, neither of the methods has a closed-form solution and resorts to a gradient-based optimization.

The inverse regression (IR) is another interesting framework for DRR. IR is based on the fact that the inverse regression, $\mathbb{E}[x|y]$, lies on the subspace spanned by $B$.

---

\(^1\)Any set of bases that spans the subspace of $B$ will be a solution.

\(^2\)Although the term *subspace* is usually meant for a linear case, however, I abuse the term for both linear and nonlinear cases throughout the paper.

\(^3\)Refer to Sec. 5.2.1 for the definition.

\(^4\)One may estimate the manifold image of the new point by extra/interpolation. However, this requires additional estimation effort.
(the bases of the central subspace), provided that the marginal distribution of \( x \) is ellipse-symmetric (e.g., a Gaussian). Thus \( B \) coincides with the principal directions in the variance of the inverse regression, namely, \( \mathbb{V}(E[x|y]) \). In [33], this variance was estimated by slicing the output space (i.e., clustering \( y \)), lending the name sliced IR (or SIR).

Despite its simplicity and a closed-form solution, SIR assumes a linear central subspace, with a strong restriction on the marginal distribution of \( x \). To cope with the limitation, a natural kernel extension (KSIR) was proposed in [70]. KSIR discovers a nonlinear central subspace and allows few restrictions on the class of distribution on \( x \), for example, admitting a nonparametric kernel density. However, KSIR still resorts to slicing of \( y \), which can result in unreliable variance estimates for high dimensional \( y \).

In this thesis I propose a novel nonlinear method for DRR that exploits the covariance functions of input as well as the output. I estimate the variance of the inverse regression under the IR framework but avoid explicit slicing by an effective use of covariance operators in RKHS. This leads to a general solution with KSIR as its special case. The proposed approach in this thesis can be reliably applied to the cases of high dimensional output, while suppressing potential noise in the output data.

The main contributions of this work address important limitations of existing DRR techniques. In particular, the approach provides the following benefits: (1) a closed-form solution, (2) a nonlinear central subspace, (3) mild assumption on the input distribution, (4) reliable estimation for high dimensional output, (5) robustness to noise, and (6) ease of generalization to new input points.

The chapter is organized as follows: I briefly review the inverse regression framework and slicing-based techniques (SIR and KSIR) in Sec. 5.1. Sec. 5.2 introduces the new approach. In Sec. 5.3 the benefits of the proposed method are demonstrated in a comprehensive set of evaluations on several regression problems.
5.1 Background

I assume that the data pair \((x \in \mathbb{R}^p, y \in \mathbb{R}^d)\) is drawn from an unknown joint distribution \(P(x, y)\), where all the expectations and (co)variances that appear in this chapter are taken with respect to \(P(x, y)\).

5.1.1 Sliced Inverse Regression (SIR)

The following theorem plays a crucial role in the IR framework. See [33] for the proof.

Without loss of generality, I assume that \(x\) is centered, i.e., \(E[x] = 0\).

**Theorem 4.** If (a) there exists a \(q\)-dim central subspace with bases \(B = [b_1, \ldots, b_q]\), i.e., \(y \perp x|B^\top x\), and (b) for any \(a \in \mathbb{R}^p\), \(E[a^\top x|B^\top x]\) is linear in \(\{b_l^\top x\}_{l=1}^q\), then \(E[x|y] \in \mathbb{R}^p\) (traced by \(y\)) lies on the subspace spanned by \(\{\Sigma_{xx}b_l\}_{l=1}^q\), where \(\Sigma_{xx}\) is the covariance of \(x\).

According to Thm. 4, \(B\) can be obtained from the \(q\) principal directions of \(E[x|y]\). That is, the column vectors of \(B\) are the \(q\) largest eigenvectors of \(V(E[x|y])\), pre-multiplied by \(\Sigma_{xx}^{-1}\). Given the data \(\{(x^i, y^i)\}_{i=1}^n\), SIR of [33] suggests to slice down (or cluster) \(y\) so as to compute the sample estimate of \(V(E[x|y])\). More specifically, after clustering \(\{y^i\}_{i=1}^n\) into \(J\) slices, \(S_1, \ldots, S_J\), and computing the slice means, \(m_j = \frac{1}{|S_j|}\sum_{i \in S_j} x^i\), to approximate \(E[x|y \in S_j]\), the sample estimate is, \(V = \sum_j p_j m_j m_j^\top\), where \(p_j = |S_j|/n\) is the \(j\)-th slice proportion.

SIR finds the directions of maximum variance, where \(n\) data points are collapsed to \(J\) slice means by affinity in \(y\) values. Not surprisingly, for the extreme case of \(J = n\), that is, when each slice is taken by a single data point, \(V\) becomes the sample covariance of \(x\), which gives rise to PCA. However, for \(J < n\), the \(y\) labels have an effect on suppressing the variance of directions within the same slice, which is a desirable strategy for the purpose of regression.

\[\exists \{\alpha_l\}_{l=0}^q \text{ s.t. } E[a^\top x|b_1^\top x, \ldots, b_q^\top x] = \alpha_0 + \sum_{l=1}^q \alpha_l \cdot b_l^\top x.\]
It is known that the condition (b) in Thm. 4 is equivalent to having an ellipse-symmetric distribution (e.g., a Gaussian) of \( x \). In summary, SIR makes two assumptions: the linearity of the central subspace and the ellipse-symmetry of the marginal distribution of \( x \). These assumptions can be strong in certain situations, and SIR would fail if the conditions are not met. In what follows, one can consider a fairly natural nonlinear extension via the RKHS mapping \( x \to \Phi(x) \), which helps relax the strong constraints of SIR.

5.1.2 Kernel Extension of Inverse Regression

In the kernel extension of SIR, \( x \) is mapped to \( \Phi(x) \in \mathcal{H}_k \), where \( \mathcal{H}_k \) is the Hilbert space induced from the kernel function \( k(\cdot,\cdot) \) defined on the \( x \) space. We assume that \( \Phi(x) \) is centered\(^6\) in \( \mathcal{H}_k \). The kernel extension consequently results in: (1) \( B \) has nonlinear basis functions \( b_l(\cdot) \in \mathcal{H}_k \), \( l = 1, \ldots, q \), (2) \( \mathbb{E}[\Phi(x)|y] \) lies on a nonlinear function space spanned by \( \{\Sigma_{xx}b_l\}_{l=1}^q \), and (3) we estimate the operator, \( \mathbb{V}(\mathbb{E}[\Phi(x)|y]) \), and its major eigenfunctions.

Similarly to SIR, KSIR of [70] estimates \( \mathbb{V}(\mathbb{E}[\Phi(x)|y]) \) by slicing the output. The KSIR algorithm is as follows:

1. Cluster \( \{y_i\}_{i=1}^n \) into \( J \) slices: \( S_1, \ldots, S_J \).

   Compute cluster means, \( m_j = \frac{1}{|S_j|} \sum_{i \in S_j} \Phi(x_i) \) for \( j = 1, \ldots, J \). \( p_j (= |S_j|/n) \) is the \( j \)-th cluster proportion.

2. Estimate the sample covariance of the slice-wise inverse regression, i.e., \( V = \sum_{j=1}^J p_j m_j m_j^\top \). Its \( q \) major eigenfunctions are denoted as \( \{v_l\}_{l=1}^q \).

3. The central subspace directions are obtained as \( b_l = \Sigma_{xx}^{-1} v_l \) for \( l = 1, \ldots, q \).

Although all the above computations can be done straightforwardly in the original input space (i.e., SIR), they cannot be represented explicitly in the RKHS. For instance,

\(^6\)The centralization of the kernel matrix is fairly straightforward and can be found in Appendix A of [48].

\(^7\)Here, I abuse the notation \( \Sigma_{xx} \) to indicate the covariance operator, and the multiplication means applying the operator to the function.
\( \mathbf{m}_j \) is a function and \( \mathbf{V} \) is an operator. However, using the well-known representer theorem [46], the eigenfunctions \( \mathbf{v} \) and the central subspace directions \( \mathbf{b} \) can be obtained in dual forms. The trick is similar to that of kernel PCA [48].

In Step-2, to solve the eigensystem \( \mathbf{V} \cdot \mathbf{v} = \lambda \cdot \mathbf{v} \), we represent \( \mathbf{v} \) as a linear combination of \( \{ \Phi(x^i) \}_{i=1}^n \), i.e., \( \mathbf{v} = \sum_{i=1}^n \alpha_i \Phi(x^i) \). Pre-multiplying by \( \Phi(x^j)^\top \) (for \( r = 1, \ldots, n \)) yields the LHS of the eigensystem as:

\[
\sum_{j=1}^J p_j (\mathbf{m}_j^\top \Phi(x^r)) \cdot \left( \sum_{i=1}^n \alpha_i (\mathbf{m}_j^\top \Phi(x^i)) \right).
\] (5.1)

Since \( \mathbf{m}_j = \frac{1}{|S_j|} \sum_{i \in S_j} \Phi(x^i) \) and \( \Phi(x)^\top \cdot \Phi(x') = k(x, x') \), stacking up \( n \) equations \( (r = 1, \ldots, n) \) results in the following dual version of the eigensystem:

\[
\mathbf{G} \mathbf{P} \mathbf{G}^\top \mathbf{\alpha} = \lambda \mathbf{K}_x \mathbf{\alpha},
\] (5.2)

where \( \mathbf{G} \) is the \((n \times J)\) matrix with \( \mathbf{G}(r, j) = \mathbf{m}_j^\top \Phi(x^r) = \frac{1}{|S_j|} \sum_{i \in S_j} k(x^i, x^r) \), \( \mathbf{P} \) is the \((J \times J)\) diagonal matrix with \( \mathbf{P}(j, j) = p_j \), \( \mathbf{K}_x \) is the kernel Gram matrix for \( \mathbf{x} \), i.e., \( \mathbf{K}_x(i, r) = k(x^i, x^r) \), and \( \mathbf{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top \). It is often the case that the eigenfunctions need to be normalized to unit-norm, which introduces extra constraints, namely, \( \mathbf{\alpha}^\top \mathbf{K}_x \mathbf{\alpha} = 1 \). Eq.(5.2) is the generalized eigenvalue problem, where we find the \( q \) major eigenvectors \( \mathbf{\alpha} \).

We note several interesting aspects of KSIR: First, the (linear) SIR can be simply derived from the KSIR algorithm with a linear kernel, \( k(x, x') = \mathbf{x}^\top \mathbf{x}' \). Secondly, similarly to the relationship between SIR and PCA, when \( J = n \), KSIR is equivalent to the kernel PCA on \( \{x^i\}_{i=1}^n \). To see this, as \( J \to n \), note that \( \mathbf{P} \to \frac{1}{n} \mathbf{I}_n \), \( (\mathbf{I}_n \) is the \((n \times n)\) identity) and \( \mathbf{G} \to \mathbf{K}_x \). Hence Eq.(5.2) reduces to \( \mathbf{K}_x \mathbf{\alpha} = n \lambda \mathbf{\alpha} \), which is the exact derivation for the kernel PCA [48].

Once we have \( \mathbf{v} \) (from the dual solution \( \mathbf{\alpha} \)), the corresponding central subspace direction \( \mathbf{b} \) of Step-3 can be obtained using a similar trick. To solve \( \Sigma_{xx} \cdot \mathbf{b} = \mathbf{v} \), we replace the covariance operator \( \Sigma_{xx} \) by the sample estimate \( \frac{1}{n} \mathbf{W}_x \mathbf{W}_x^\top \), where \( \mathbf{W}_x = [\Phi(x^1), \ldots, \Phi(x^n)] \). By letting \( \mathbf{b} = \sum_{i=1}^n \beta_i \Phi(x^i) \), and pre-multiplying by \( \Phi(x^r)^\top \) (for \( r = 1, \ldots, n \)), we have the closed-form solution:

\[
\beta = n \mathbf{K}^{-1}_x \mathbf{\alpha},
\] (5.3)
where $\beta = [\beta_1, \ldots, \beta_n]^\top$.

The nonlinear central subspace is then represented by $q$ basis functions, $\{b_l\}_{l=1}^q$ from the dual solutions $\{\beta_l\}_{l=1}^q$. For a new test input point $x^*$, its low-dimensional representation $z^* \in \mathbb{R}^q$ can be obtained by projecting $\Phi(x^*)$ onto the central subspace. That is, the $l$-th element of $z^*$ is, $z_{l}^* = b_l^\top \Phi(x^*) = k_{l}^\top \beta_l$, for $l = 1, \ldots, q$, where $k_{l} = [k(x^1, x^*), \ldots, k(x^n, x^*)]^\top$.

The kernel extension of inverse regression resolves certain limitation of the (linear) SIR. Not restricted to a linear central subspace, it allows the distribution of $x$ to be more generous. However, KSIR’s slicing-based estimation of $\mathbb{V}(\mathbb{E}[\Phi(x)|y])$ would be unreliable for high-dimensional $y$. This makes KSIR restricted to single-output regression or classification settings only [70].

In the following section, I propose a novel estimation method that avoids slicing by exploiting the kernel matrices of input and output. As we will see in Sec. 5.3, the approach is successfully applied to regression problems with a large number of output variables that often arise in computer vision areas (e.g., human figure pose estimation, image reconstruction with noise removal).

5.2 Proposed Approach

5.2.1 IR using Covariance Operators in RKHS

In the new approach the estimation of $\mathbb{V}(\mathbb{E}[\Phi(x)|y])$ is based on the (cross) covariance operator theorems [3, 12, 61]. First, I introduce the covariance operator as a natural RKHS extension of the covariance matrix in the original space. For two random vectors $y$ and $x$ endowed with Hilbert spaces $\mathcal{H}_y$ with $k_y(\cdot, \cdot)$ and $\mathcal{H}_x$ with $k_x(\cdot, \cdot)$, respectively, I define $\Sigma_{yx} := \text{Cov}(\Phi(y), \Phi(x))$, namely, the (cross) covariance in the feature spaces. Note that $\Sigma_{yx}$ is an operator that maps from $\mathcal{H}_x$ to $\mathcal{H}_y$, thus having dimension $\text{dim}(\mathcal{H}_y) \times \text{dim}(\mathcal{H}_x)$. One can similarly define other covariance operators, $\Sigma_{xy}, \Sigma_{yy}, \text{or } \Sigma_{xx}$.

---

\^8A more precise definition would be: for $\forall g \in \mathcal{H}_y$ and $\forall f \in \mathcal{H}_x$, $\langle g, \Sigma_{yx} f \rangle = \mathbb{E}[(g(y) - \mathbb{E}g(y))(f(x) - \mathbb{E}f(x))]$. 
For notational convenience, I treat the covariance operators as if they were matrices. For instance, for $g \in \mathcal{H}_y$ and $f \in \mathcal{H}_x$, $g^\top \Sigma_{yx}f$ means the inner product $\langle g, \Sigma_{yx}f \rangle$ in $\mathcal{H}_y$ space. Besides, $f^\top \Phi(x)$ and $\Phi(x)$ are used interchangeably, as they are equivalent from the Riesz representation theorem [65]. Now I define the conditional covariance operator of $y$ given $x$, denoted by $\Sigma_{yy|x}$, as:

$$\Sigma_{yy|x} := \Sigma_{yy} - \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy}. \quad (5.4)$$

The following theorem (also found in [12]) states that under some mild condition, $\Sigma_{yy|x}$ equals to the expected conditional variance of $\Phi(y)$ given $x$ (i.e., $E[V(\Phi(y)|x)]$). See Appendix at the end of the chapter for the proof.

**Theorem 5.** For any $g \in \mathcal{H}_y$, if there exists $f \in \mathcal{H}_x$ such that $E[g(y)|x] = f(x)$, then $\Sigma_{yy|x} = E[V(\Phi(y)|x)]$.

The condition in Thm. 5 implies that the regression function from $x$ to $g(y)$ for any given $g \in \mathcal{H}_y$ has to be linear in RKHS, namely, of the form $f^\top \Phi(x)$ for some $f \in \mathcal{H}_x$. However, this is rather a reasonable condition as it corresponds to a rich family of smooth functions in the original space (i.e., $f^\top \Phi(x) = f(x)$) [47].

Now I propose to represent $V(E[\Phi(x)|y])$ of IR in terms of the conditional covariance operators. More specifically, using the well-known $E-V-V-E$ identity\(^9\), it can be written as:

$$V(E[\Phi(x)|y]) = V(\Phi(x)) - E[V(\Phi(x)|y)]. \quad (5.5)$$

From Thm. 5, the second term of the RHS in Eq.(5.5) equals to $\Sigma_{xx|y}$ (changing the role of $x$ and $y$ in Eq.(5.4)), assuming that the inverse regression, $E[f(x)|y]$, is a smooth function of $y$ for any $f \in \mathcal{H}_x$ (i.e., $\exists g \in \mathcal{H}_y$ s.t. $E[f(x)|y] = g(y)$). As $V(\Phi(x)) = \text{Cov}(\Phi(x), \Phi(x)) = \Sigma_{xx}$, we have:

$$V(E[\Phi(x)|y]) = \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx}. \quad (5.6)$$

Given the data $\{(x^i, y^i)\}_{i=1}^n$, the sample estimate of Eq.(5.6) can be written as $
abla(\Phi(x)|y) = \hat{\Sigma}_{xy} \hat{\Sigma}_{yy}^{-1} \hat{\Sigma}_{yx}$. The sample covariance operators ($\hat{\Sigma}$) can be estimated

\(^9\) $V(Y) = E[V(Y|X)] + V(E[Y|X])$ for any $X, Y$. 

similarly as the sample covariance matrices. For instance, \( \hat{\Sigma}_{xy} = \frac{1}{n} W_x W_y^T \), where 
\[
W_x = [\Phi(x^1), \ldots, \Phi(x^n)] \quad \text{and} \quad W_y = [\Phi(y^1), \ldots, \Phi(y^n)].
\]
Then \( \hat{\mathcal{Y}}(\mathbb{E}[\Phi(x)|y]) \) is:
\[
\left( \frac{1}{n} W_x W_y^T \right) \left( \frac{1}{n} (W_y W_y^T + n\epsilon I) \right)^{-1} \left( \frac{1}{n} W_x W_x^T \right)
= \frac{1}{n} W_x W_y^T (W_y W_y^T + n\epsilon I)^{-1} W_y W_x^T
= \frac{1}{n} W_x W_y^T W_y (W_y W_y^T + n\epsilon I_n)^{-1} W_x^T
= \frac{1}{n} W_x \mathbb{K}_y (\mathbb{K}_y + n\epsilon I_n)^{-1} W_x^T.
\]
(5.7)

Here \( I \) is the \((\dim(H_y) \times \dim(H_y))\) identity operator, and \( I_n \) is the \((n \times n)\) identity matrix. Note that a small positive \( \epsilon \) is added to the diagonals of \( W_y W_y^T \) so as to circumvent potential rank deficiency in estimating \( \Sigma_{yy} \) and its inverse. As will be discussed in Sec. 5.2.2, \( \epsilon \) plays a crucial role as a kernel regularizer in smoothing the affinity structure of \( y \). In Eq.(5.7), I use the fact that \( W_y (W_y W_y^T + n\epsilon I_n) = (W_y W_y^T + n\epsilon I) W_y \). In Eq.(5.8), \( \mathbb{K}_y = W_y^T W_y \) is the \((n \times n)\) kernel matrix on \( y \), i.e., \( \mathbb{K}_y(i, r) = k_y(y^i, y^r) \).

The eigenfunctions of Eq.(5.8) can be obtained by pre-multiplying the eigensystem, 
\( \hat{\mathcal{Y}}(\mathbb{E}[\Phi(x)|y]) \cdot v = \lambda \cdot v \), by \( W_x^T \). From 
\[
v = \sum_{i=1}^{n} \alpha_i \Phi(x^i) = W_x \alpha,
\]
we have:
\[
\frac{1}{n} \mathbb{K}_y (\mathbb{K}_y + n\epsilon I_n)^{-1} \mathbb{K}_x \alpha = \lambda \alpha,
\]
(5.9)
where \( \mathbb{K}_x = W_x^T W_x \) is the kernel matrix on \( x \). Once we get \( v \), the Step-3 of KSIR follows to find \( b \). From now on, I denote the inverse regression technique described in this section by COIR (i.e., Covariance Operator IR).

COIR has a closed-form solution (Eq.(5.9)) to the nonlinear central subspace, while making little assumption on the input distribution. Noticeably, COIR avoids KSIR’s slicing-based estimation by incorporating a smooth output kernel. This makes COIR not only handle high-dimensional output reliably, but also robust to potential noise in the output data. Furthermore, I show that KSIR is a special case of COIR.

5.2.2 KSIR as a Special Case of COIR

Recall from KSIR that \( P \) is the \((J \times J)\) cluster proportion diagonal matrix with the \( j \)-th element, \( p_j = |S_j|/n \). I let \( C \) be the \((n \times J)\) cluster indicator 0/1 matrix whose \( i \)-th
row has all 0's but 1 at the \( j \)-th position where \( i \in S_j \). Noticing that \( G = \frac{1}{n} K_x C P^{-1} \), the KSIR equation in Eq.(5.2) can be written as:

\[
\frac{1}{n} K_x (\frac{1}{n} C P^{-1} C^\top) K_x \alpha = \lambda K_x \alpha.
\]  

(5.10)

On the other hand, the COIR equation in Eq.(5.9) is (after pre-multiplying by \( K_x \)):

\[
\frac{1}{n} K_x (K_y (K_y + n \epsilon I_n)^{-1}) K_x \alpha = \lambda K_x \alpha.
\]  

(5.11)

From Eq.(5.10) and Eq.(5.11), the equivalence between KSIR and COIR is made by:

\[
K_y (K_y + n \epsilon I_n)^{-1} = \frac{1}{n} C P^{-1} C^\top.
\]  

(5.12)

Now consider an ideal case that the output data \( \{y^i\}_{i=1}^n \) are collapsed to \( J \) distinct points that are infinitely far apart from one another. (Without loss of generality, the data points are arranged to be grouped in the same clusters.) Assuming an RBF kernel, this in turn makes \( K_y \) a block 0/1 matrix, where each block of 1’s corresponds to each of \( J \) clusters. For instance, when \( n = 6, J = 3, |S_1| = 3, |S_2| = 1, \) and \( |S_3| = 2 \),

\[
K_y = \begin{bmatrix}
E_3 & 0 & 0 \\
0 & E_1 & 0 \\
0 & 0 & E_2
\end{bmatrix},
\]  

(5.13)

where \( E_m \) denotes the \((m \times m)\) matrix with all 1’s.

I show that under the above assumption, Eq.(5.12) is indeed true when \( \epsilon \to 0 \). For the block 0/1 Gram matrix \( K_y \) (e.g., Eq.(5.13)), the LHS of Eq.(5.12) can be expressed as:

\[
K_y (K_y + n \epsilon I_n)^{-1} = \begin{bmatrix}
c_1 E_{|S_1|} & 0 & \cdots \\
0 & \cdots & \cdots \\
0 & c_J E_{|S_J|}
\end{bmatrix},
\]  

(5.14)

where \( c_j = \frac{1}{|S_j| + n \epsilon} \). One can easily verify this by post-multiplying both sides of Eq.(5.14) by \((K_y + n \epsilon I_n)\). It is also straightforward to rewrite the RHS of Eq.(5.12) as:

\[
\frac{1}{n} C P^{-1} C^\top = \begin{bmatrix}
\frac{1}{np_1} E_{|S_1|} & 0 & \cdots \\
0 & \cdots & \cdots \\
0 & \frac{1}{np_J} E_{|S_J|}
\end{bmatrix}.
\]  

(5.15)
Therefore, Eq.(5.12) reduces to:

$$|S_j| + n\epsilon = np_j, \quad \text{for } j = 1, \ldots, J.$$  \hfill (5.16)

As $\epsilon \to 0$, Eq.(5.16) implies that $p_j = |S_j|/n$, which is exactly the maximum likelihood (ML) estimate of the cluster proportion employed by KSIR. That is, KSIR is a special case of COIR having 0/1 Gram matrix $K_y$ (from the assumed $J$-collapsed perfect clustering) with a vanishing $\epsilon$. For a non-negligible $\epsilon$, the equivalence turns into $p_j = |S_j|/n + \epsilon$, where $\epsilon$ now serves as a regularizer (or a smoothing prior) in the ML estimation.

For a general (non-0/1) kernel matrix $K_y$, COIR can be naturally viewed as a smoothed extension of KSIR. In a nutshell, COIR exploits the kernel structure of the output space by the effective use of covariance operators in RKHS, where $\epsilon$ acts as a kernel regularizer.

### 5.3 Empirical Evaluation

I demonstrate the benefits of COIR against the existing DRR techniques, SIR and KSIR. I also compare it with the unsupervised dimension reduction technique, the kernel PCA (denoted by KPCA)\(^\text{10}\) to illustrate the advantage of DRR. Unless stated otherwise, the kernel-based methods (i.e., COIR, KSIR, and KPCA) employ the RBF kernel. SIR and KSIR use the k-means clustering algorithm for output slicing.

#### 5.3.1 Synthetic Curves Dataset

The dataset called curves was devised for testing KSIR in [70], where it is generated by:

$$y = \text{sign}(b_1^T x + \epsilon_1) \cdot \log(\|b_2^T x + a_0 + \epsilon_2\|)$$

for some $b_1, b_2 \in \mathbb{R}^{15}$, $x \sim \mathcal{N}(0, I_{15})$, $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, 1)$, and a constant $a_0$. The input is 15-dim, but the central subspace is at most 2-dim as $y$ is decided by $\{b_1^T x\}_{l=1}^2$. The output $y$ is 1-dim. For 300 data points generated, I plotted the 2D central subspaces estimated by the competing methods in Fig. 5.1. In the plots, each point is colored by its true $y$ value, depicting higher values as

\(^{10}\text{Although we do not present other unsupervised methods here, their results are not significantly different from KPCA.}\)
warmer (reddish) and lower as cooler (bluish). For SIR and KSIR, we vary \( J \) (#slices), where the extreme case of \( J = n \) gives rise to PCA and KPCA, respectively.

In SIR (regardless of \( J \)), the data points are roughly grouped into 4 clusters by \( y \) values: red, yellow, green, and blue. However, the partial overlap of points in red (higher \( y \)) and blue (lower \( y \)) can result in significant error in the regression estimation based on it. KSIR is more sensitive to the choice of \( J \). When \( J = 5 \), the clusters are separated better, but mixed within each cluster. Increasing \( J \) resolves mixing, but the clusters get closer to one another. On the other hand, COIR (Fig. 5.1(g)) exhibits smooth and clear discrimination of data along the \( y \) values. Moreover, COIR is not involved in choosing the parameter \( J \), a sensitive task required by KSIR. The unsupervised PCA and KPCA produce random clutter since they simply project the isotropic Gaussian data onto a 2D plane with no information about \( y \).

To simulate the noisy nature of real-world data, I devise an interesting setting by adding 4 Gaussian white noise dimensions to \( y \) (5-dim in total). The results of KSIR and COIR are shown in Fig. 5.1(h) and Fig. 5.1(i), respectively, colored by true (noise-removed) \( y \) values. In KSIR, due to the clustering error from the noise, each cluster has data points mixed with different \( y \) values (e.g., red, yellow, and green points in the same cluster). However, COIR still lays out the data along the \( y \) values (from blue/left to red/right), and even a simple linear regressor (e.g., linear in the X axis) seemingly works well. COIR’s robustness to noise originates from its utilization of the smooth output kernel.

### 5.3.2 Head Pose Estimation

The dataset\(^\text{11}\) consists of 500 face images of size \((64 \times 64)\), rendered from different views with varying lighting direction. The task is to predict the pose (horizontal and vertical rotation angles) and/or the lighting direction from the 4096-dim image. I consider two regression settings by selectively taking the regression output variables of interest: (1) suppressing the lighting variable (i.e., \( y \) is 2D pose) and (2) including all (i.e., \( y \) is 3D

\(^{11}\text{Available at http://isomap.stanford.edu/datasets.html.}\)
Figure 5.1: Central subspaces for the curves dataset.
(pose + lighting)). By taking into account the degrees of freedom in the output, the central subspace dimensions are accordingly chosen as 2 and 3 for the setting-(1) and (2), respectively. I show the central subspaces estimated by COIR and KSIR in Fig. 5.2 for the setting-(1), and Fig. 5.3 for the setting-(2).

Here, the data points are randomly partitioned into 80%/20% train/test sets. (In the figures, the test points are depicted in black circles.) In Fig. 5.3, to visualize the 3D central subspace data points in 3D output, I rotate the axes to be seen as discriminatively as possible. In both settings, COIR lays out the data almost linearly along the output values with good generalization. In KSIR, however, the data points are often mixed significantly. In particular, the red points are intermingled with the green points in Fig. 5.2(e), while the points are overall mixed in Fig. 5.3(e).

5.3.3 Noisy Sparse Head Pose Estimation

I consider a more realistic setting for the head pose estimation. \( n = 683 \) images are collected, comprising faces of about 100 subjects instructed to move their heads in arbitrary rotation. A standard face detector was applied (with a manual refinement) to locate a tight bounding box around the face. To suppress the in-plane head tilt, the face is aligned in up-front. The image is further resized to \((80 \times 80)\), a 6400-dim input vector.

For the output, the approximate out-of-plane rotation angles along X and Y axes are recorded. Note that this introduces substantial amount of noise in the output data. I denote the 2D output by \( y = [y_1, y_2]^\top \), where \( y_1 \) and \( y_2 \) are vertical and horizontal rotation angles, respectively. In addition, the data is sparse in the \( y \) space as most of the data points (about 90%) have one of the angles equal to 0 (i.e., purely horizontal or vertical movement).

Fig. 5.4 shows the 2D central subspaces estimated by the competing nonlinear methods\(^\text{12}\). For visualization, we colored the dim-reduced input point by each of \( y_1 \) and \( y_2 \) (e.g., Fig. 5.4(a) and Fig. 5.4(d) depict the same points for COIR, but colored by \( y_1 \)).

\(^{12}\)We skip the (linear) SIR result as its performance is worse than KSIR.
Figure 5.2: 2D central subspaces of face images with the output comprised of 2D pose. The points are colored by the true Left/Right pose angles in (a)/(d), Up/Down in (b)/(e), and (c)/(f) shows the face images superimposed. The test points are depicted in black circles.
Figure 5.3: 3D central subspaces of face images with the output comprised of 2D pose and the lighting. The points are colored by the true Left/Right pose angles in (a)/(d), Up/Down in (b)/(e), and the lighting direction in (c)/(f). The test points are depicted in black circles.
and $y_2$, respectively. We see that COIR lays out the data points along the head pose quite obviously, where X and Y axes roughly correspond to horizontal ($y_2$) and vertical ($y_1$) angles, respectively. On the other hand, KSIR exhibits severe ambiguity in $y_1$. For COIR, we also superimpose some of the held-out test face images (whose $y$ are not known to the algorithm) in Fig. 5.5. Despite the sparsity and the noise in the output data, COIR generalizes well even for the combined rotation (e.g., the face image at around $(40, 70)$).

### 5.3.4 Human Figure Pose Estimation

Another interesting problem is to estimate the human body pose from a silhouette image. It is advantageous to apply DRR techniques as one often wants to find the
intrinsic low-dimensional (2D or 3D as is widely believed) representation of the input image in predicting the body pose.

I use the walking sequence of length $\sim 400$ (containing about 3 walking cycles) obtained from the CMU motion capture database. The output $y$ is composed of 59 3D joint angles at 31 articulation points. The input $x$ is the silhouette image of size $(160 \times 100)$ taken at a side view. As the silhouette image is completely ambiguous in discriminating two opposite poses with left/right arms/legs switched, I focus on a half cycle as shown in Fig. 5.6. In the figure, the skeleton images are drawn using the 3D joint angles.

Trained on the first 80% of the frames, the 2D central subspaces estimated by COIR, KSIR, and KPCA are shown in Fig. 5.7. COIR yields a circular trajectory unambiguous within a half cycle. On the other hand, KSIR is distorted at the beginning/end of the half cycle. Especially, the points at $t = 62$ and $t = 110$ adjoin each other too closely, which would result in a large estimation error in pose prediction. This illustrates that KSIR’s slicing-based estimation is unreliable for high-dimensional output. Note that the unsupervised KPCA shows a much severer distortion than KSIR. Moreover, COIR
Figure 5.6: Selected skeleton and silhouette images for a half walking cycle: From stand-up ($t = 56$), right-leg forwarding, left-leg catching-up, and back to stand-up ($t = 120$). The skeleton images are drawn using the 3D joint angles.

<table>
<thead>
<tr>
<th>Input Space</th>
<th>COIR</th>
<th>KSIR</th>
<th>KPCA</th>
<th>Image $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Regression</td>
<td>6.1782</td>
<td>8.9054</td>
<td>8.6592</td>
<td>6.5151</td>
</tr>
<tr>
<td>GP Regression</td>
<td>5.8632</td>
<td>8.1602</td>
<td>8.5436</td>
<td>5.9535</td>
</tr>
</tbody>
</table>

Table 5.1: Test (RMS) errors in 3D human figure pose estimation.

generalizes well for the test (red) points.

The actual regression estimation is also conducted. I employ two most popular regression methods: the nearest neighbor (NN) and the Gaussian Process (GP) regression\textsuperscript{13} \cite{68}. See Table 5.1 for the test RMS errors. Even though the regressors are built on the dimension-reduced (2D) input space, the prediction performance of COIR is never worse than that based on the silhouette image $x$ itself as input. On the other hand, dimension reduction by KSIR (or KPCA) entails significant loss of information in predicting output.

5.3.5 Hand-written Digit Image Reconstruction

I devise the image reconstruction experiment with the USPS hand-written digit images \cite{32}. By adding random scratch lines with varying thickness and orientation on the normalized ($16 \times 16$) digit images, the task is to denoise or reconstruct the image. So, the regression problem is to predict the original unscratched image (output $y$) from the scratched image (input $x$). Both $y$ and $x$ are of 256-dim.

\textsuperscript{13}For the multiple output regression, we assume independent GP priors, which results in independent GP prediction for each output dimension \cite{31}.
Figure 5.7: Central subspaces for silhouette images from walking motion: The blue (red) points indicate train (test) data points.
Table 5.2: Test (RMS) errors in scratched digit image denoising.

<table>
<thead>
<tr>
<th>Input Space</th>
<th>COIR</th>
<th>KSIR</th>
<th>Image $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Regression</td>
<td>8.5334</td>
<td>11.4909</td>
<td>9.3605</td>
</tr>
<tr>
<td>GP Regression</td>
<td>8.1454</td>
<td>10.7259</td>
<td>9.1036</td>
</tr>
</tbody>
</table>

From the database, I use a subset of 2000 images for training and another 2000 for testing. The central subspace dimension is chosen as 30. The test reconstruction (denoising) RMS errors are shown in Table 5.2, while some of the denoised test images by the NN regression are depicted in Fig. 5.8. We can see that COIR is robust to noise with improved prediction accuracy compared to the regression based on the image input itself. KSIR again suffers from unreliable slicing-based estimation in the high-dimensional output space.
<table>
<thead>
<tr>
<th>Input Space</th>
<th>COIR</th>
<th>KSIR</th>
<th>SIR</th>
<th>KPCA</th>
<th>Original x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>1.8925</td>
<td>1.9383</td>
<td>9.5191</td>
<td>10.0073</td>
<td>9.5191</td>
</tr>
</tbody>
</table>

Table 5.3: Temperature map: RMS errors (in K) by linear regression.

5.3.6 Temperature Map

I test COIR on the global temperature data obtained from the satellite measurement of temperatures in the middle troposphere. It provides about 300 temperature maps, one for each month since April 1978. I take the map of December 2004 (See Fig. 5.9(a)) which was also used in the manifold KDR [38]. The map is a (72 × 144) matrix, where each element has a temperature value (in K) at its position indexed by (latitude, longitude). Hence, the regression problem is: \( y = \text{temperature (1D)} \) and \( x = (\text{latitude, longitude}) \) (2D). I randomly split the data into 80%/20% train/test sets.

Note that the 1D central subspace is nonlinear due to the ellipsoidal manifold structure of the input space. To see this, refer to Table 5.3, where the linear regression on the original input space results in much larger prediction error than those based on the nonlinear central subspaces (e.g. COIR or KSIR).

As shown in Fig. 5.9, the nonlinear DRR techniques (both COIR and KSIR) perform well, discovering a linear relationship between the output and the input projected on to the central subspace. Note that KSIR’s slicing-based estimation is reliable for this 1D output case. As expected, the linear SIR and the unsupervised KPCA simply fail (i.e., the mappings from the dimension-reduced input to \( y \) are one-to-many as shown in Fig. 5.9(d) and Fig. 5.9(e)).

See Table 5.3 for the train/test RMS errors using the linear regression estimation on the dimension-reduced input space. The nonlinear central subspaces (COIR and KSIR) result in significantly better performance than others (including the regression from the original space), while COIR marginally outperforms KSIR.

---

Figure 5.9: Temperature map: (a) is the true $y$ (temperature) map. (b)-(e) plot $y$ vs. dimension-reduced $x$ for the competing methods (train/test points are in blue/red). (f)-(i) show the predicted temperatures using the linear regression estimation on the central subspaces. (j)-(m) show the difference maps between the true and the predicted temperatures (the larger difference, the darker).
Appendix: Proof of Thm. 5

To prove Thm. 5, one may need the following lemma which states that $\Sigma_{yy|x}$ is tightly related to the optimal linear (in RKHS) regressor in terms of the variance of the error. More specifically, when we regress from $x$ to $g(y)$ for a given $g \in \mathcal{H}_y$, the prediction error cannot be smaller than $g^\top \Sigma_{yy|x} g$.

**Lemma 6.** For any $g \in \mathcal{H}_y$, $\inf_{f \in \mathcal{H}_x} \mathbb{V}(g(y) - f(x)) = \inf_{f \in \mathcal{H}_x} \mathbb{V}(g^\top \Phi(y) - f^\top \Phi(x)) = g^\top \Sigma_{yy|x} g$.

**Proof.** From the co-linearity of Cov$(\cdot, \cdot)$, $\mathbb{V}(g^\top \Phi(y) - f^\top \Phi(x)) = \text{Cov}(g^\top \Phi(y) - f^\top \Phi(x), g^\top \Phi(y) - f^\top \Phi(x)) = g^\top \Sigma_{yy} g - 2f^\top \Sigma_{xy} g + f^\top \Sigma_{xx} f$. As the latter is quadratic (convex) in $f$, by taking the gradient to 0, namely, $\partial f = -2 \Sigma_{xy} g + 2 \Sigma_{xx} f = 0$, its infimum is found at $f^* = \Sigma_{xx}^{-1} \Sigma_{xy} g$. Plugging $f^*$ back yields $g^\top \Sigma_{yy|x} g$. $\square$

From Lemma 6 and the well-known $E-V-V-E$ identity, we have:

$$g^\top \Sigma_{yy|x} g = \inf_{f \in \mathcal{H}_x} \mathbb{V}(g(y) - f(x)) = \inf_{f \in \mathcal{H}_x} \left\{ \mathbb{E}[\mathbb{V}(g(y) - f(x)|x)] + \mathbb{V}(\mathbb{E}[g(y) - f(x)|x]) \right\} = \mathbb{E}[\mathbb{V}(g(y)|x)] + \inf_{f \in \mathcal{H}_x} \mathbb{V}(\mathbb{E}[g(y)|x] - f(x)).$$

Note that the second term is non-negative. From the assumption, as there always exists $f \in \mathcal{H}_x$ that makes the second term 0, $g^\top \Sigma_{yy|x} g = \mathbb{E}[\mathbb{V}(g(y)|x)] = g^\top \mathbb{E}[\mathbb{V}(\Phi(y)|x)] g$ for any $g \in \mathcal{H}_y$. This completes the proof.
Chapter 6
Conclusions

In this thesis, I addressed two challenging tasks related to regression: structured output regression and dimensionality reduction.

For the structured output regression problems, I proposed novel discriminative approaches for the probabilistic dynamic models, and empirically showed that the conditional modeling and the discriminative training of generative models can yield accurate and computationally efficient state estimation. The proposed algorithms often achieved even superior prediction performance to that of the complex nonlinear dynamic models.

In particular, I addressed the integrability issue of the discriminative model in the real state domain, by casting the learning problem as a convex optimization subject to a convex constrained feasible parameter set. Moreover, the new inference algorithm is faster and numerically more stable than the Kalman smoothing, which allows us to incorporate a large number of measurement features.

Similar approaches may be applicable to a wide range of other problems that arise in AI and related fields. For instance, denoising of time series or images, prediction of financial trends are but few of the structured regression problems that can leverage the increased accuracy of the newly proposed framework. Some of these directions will be explored in the future work. Furthermore, I plan to extend the methods to piecewise linear models such as the switching dynamic model (e.g., [40]) which can handle composite motions of different styles and types.

The dimensionality reduction for regression (DRR), a supervised framework with a real multivariate label, is very useful for visualizing high-dimensional data, efficient regressor design with a reduced input dimension, and elimination of noise in input data by uncovering the essential information for predicting output. I introduced a novel
nonlinear method for DRR that exploits the kernel matrices of both input and output using the covariance operators in RKHS. In a comprehensive set of evaluations, I demonstrated that the approach can successfully discover central subspaces reliably and robustly for high dimensional noisy data. In future work, I will address the computational issue of inverting a kernel matrix for a large dataset. I also plan to extend the framework to a semi-supervised setting to take advantage of less expensive unlabeled data.
References


Vita

Minyoung Kim

2003-2008  Ph.D. in Computer Science, Rutgers University
1998-2000  M.S. in Computer Engineering from Seoul National University
1994-1998  B.A. in Computer Engineering from Seoul National University
2005-2008  Graduate Assistant, Computer Science, Rutgers University
2003-2005  Teaching Assistant, Computer Science, Rutgers University

Publications


• **Minyoung Kim** and Vladimir Pavlovic, “Discriminative Learning of Mixture of Bayesian Network Classifiers for Sequence Classification”, *Computer Vision and Pattern Recognition (CVPR)*, 2006.


• **Minyoung Kim** and Vladimir Pavlovic, “Efficient Discriminative Learning of Mixture of Bayesian Network Classifiers for Sequence Classification”, *Snowbird Learning Workshop*, 2006.