

# Acknowledgment

**Funding Support.** This book is primarily based on research results that have been developed since 2018. Thanks to the generous startup funds from UC Berkeley (2018) and the University of Hong Kong (2023), Yi Ma was able to embark and focus on this new exciting research direction in the past eight years or so. Through these years, for research in this direction, Yi Ma and his research team at Berkeley have been supported by the following research grants:

- The multi-university *THEORINET* project for the Foundations of Deep Learning, jointly funded by the Simons Foundation and the National Science Foundation (DMS grant #2031899);
- The *Closed-Loop Data Transcription via Minimizing Rate Reduction* project funded by the Office of Naval Research (grant N00014-22-1-2102);
- The *Principled Approaches to Deep Learning for Low-dimensional Structures* project funded by the National Science Foundation (CISE grant #2402951).

This book would have not been possible without the financial support for these research projects. The authors have drawn tremendous inspiration from research results by colleagues and students who have been involved in these projects.



# Notation

In this book we employ notation from a variety of mathematical disciplines, summarized below. The reader is encouraged to consult this list as a reference as they read the various chapters of the book. We have tried to keep their usage as coherent as possible across chapters.

As a first example, for a natural number  $n$  the set  $\{1, 2, \dots, n\}$  is denoted  $[n]$ .

## Linear Algebra Notation

- Scalars (either deterministic or random) are non-bold, e.g.,  $x$ ,  $X$ .
- Vectors (either deterministic or random) are lower-case bold, e.g.,  $\mathbf{x}$ ,  $\boldsymbol{\pi}$ . Entries of vectors are denoted  $x_i$  or  $\pi_i$ , alternatively  $(\mathbf{x})_i$ . Generic terms (which can be scalars, vectors, matrices, or higher-order tensors) are also lower-case bold.
- Matrices (either deterministic or random) are capital bold, e.g.,  $\mathbf{X}$ ,  $\boldsymbol{\Pi}$ . Entries of matrices are denoted  $X_{ij}$  or  $\Pi_{ij}$ , alternatively  $(\mathbf{X})_{ij}$ . Columns of matrices are lower-case bold vectors, e.g.,  $\mathbf{x}_i$  is the  $i^{\text{th}}$  column of  $\mathbf{X}$ , unless otherwise defined.
- The transpose of a matrix is  $\top$ , e.g.,  $\mathbf{A}^\top$ . The adjoint is  $*$ , e.g.,  $\mathbf{A}^*$ . The pseudo-inverse is  $\dagger$ , e.g.,  $\mathbf{A}^\dagger$ .
- The rank of a matrix is  $\text{rank}(\mathbf{A})$ , the trace is  $\text{tr}(\mathbf{A})$ , the determinant is  $\det(\mathbf{A})$ , and the log-determinant is  $\text{logdet}(\mathbf{A})$ .
- The element-wise multiplication of matrices is  $\mathbf{A} \odot \mathbf{B}$ . Element-wise squaring is  $\mathbf{A}^{\odot 2}$ , etc. Similarly, the Kronecker product of matrices is  $\mathbf{A} \otimes \mathbf{B}$ . The iterated Kronecker product is  $\mathbf{A}^{\otimes 2}$ , etc.
- For a function  $f: \mathbb{R} \rightarrow \mathbb{R}$ , its element-wise application to a matrix  $\mathbf{X}$  is  $f[\mathbf{X}]$ , i.e., with square brackets. For a function  $f: \mathbb{R}^d \rightarrow \mathbb{R}^k$  and a matrix  $\mathbf{X} \in \mathbb{R}^{d \times n}$ , we may broadcast  $f$  to apply to each column without special notation, i.e.,  $f(\mathbf{X}) = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)] \in \mathbb{R}^{k \times n}$ .

- The (Euclidean) unit sphere is  $\mathbb{S}^{n-1} \subseteq \mathbb{R}^n$ . The (Euclidean) unit ball is  $\mathbb{B}^n \subseteq \mathbb{R}^n$ .
- The set of orthogonal matrices are  $\mathbf{O}(m, n) \subseteq \mathbb{R}^{m \times n}$  or  $\mathbf{O}(n) = \mathbf{O}(n, n)$ .
- Symmetric matrices are  $\mathbf{Sym}(n)$ , symmetric PSD matrices are  $\mathbf{PSD}(n)$ , symmetric PD matrices are  $\mathbf{PD}(n)$ .
- The eigenvalues of symmetric matrix  $\mathbf{S} \in \mathbf{Sym}(n)$  are all real by the spectral theorem; they are denoted  $\lambda_i(\mathbf{S})$  and ordered such that  $\lambda_1(\mathbf{S}) \geq \dots \geq \lambda_n(\mathbf{S})$ .
- The singular values of  $\mathbf{A} \in \mathbb{R}^{m \times n}$  with  $\text{rank}(\mathbf{A}) = r$  are denoted  $\sigma_i(\mathbf{A})$  and ordered such that  $\sigma_1(\mathbf{A}) \geq \dots \geq \sigma_r(\mathbf{A}) > 0$ . By convention we set  $\sigma_{r+1}(\mathbf{A}) = \sigma_{r+2}(\mathbf{A}) = \dots = 0$ .
- The projection onto a set  $\mathcal{K}$  is denoted  $\text{proj}_{\mathcal{K}}$ .
- The  $\ell^p$  norms of vectors are denoted by  $\|\mathbf{x}\|_p$ ,  $p \in [0, \infty]$ .
- The Euclidean operator norm on matrices is  $\|\mathbf{A}\| = \sigma_1(\mathbf{A})$ .
- The Frobenius norm on matrices is  $\|\mathbf{A}\|_F = \text{tr}(\mathbf{A}^\top \mathbf{A})$ .
- The Euclidean inner product of vectors is  $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{y}^\top \mathbf{x}$ .
- The Frobenius inner product on matrices is  $\langle \mathbf{X}, \mathbf{Y} \rangle = \text{tr}(\mathbf{Y}^\top \mathbf{X})$ .
- The all-ones vector/matrix is denoted  $\mathbf{1}$  (the shape should be obvious from context). Similarly, the all-zeros vector/matrix is denoted  $\mathbf{0}$ . The identity matrix is denoted  $\mathbf{I}$ .

## Probability Notation

- The base probability measure is  $\mathbb{P}$ , i.e., the probability of an event  $A$  occurring is  $\mathbb{P}[A]$ . We may specify the distribution of a random variable using a subscript, i.e.,  $\mathbb{P}_{\mathbf{x} \sim \mu}[\mathbf{x} \in S]$  describes the probabilities associated to a random variable  $\mathbf{x}$  with distribution (measure)  $\mu$ . If two random variables  $\mathbf{x}$  and  $\mathbf{x}'$  have the same distribution, we write  $\mathbf{x} \stackrel{\text{d}}{=} \mathbf{x}'$ .
- The expected value operator is  $\mathbb{E}$ , with the same subscript caveat.
- The covariance operator is  $\text{Cov}$ , with the same subscript caveat.
- The correlation operator is  $\text{Corr}$ , with the same subscript caveat.
- In certain probabilistic modeling contexts where a density is assumed (especially in Chapters 3 and 7), we also use the notation  $p_{\mathbf{x}}$  for the density of a random variable  $\mathbf{x}$ . This extends to conditional densities, e.g.  $p_{\mathbf{x}|\mathbf{y}}$  for the density of  $\mathbf{x}$  conditioned on  $\mathbf{y}$ .

- We generally use Greek letters to denote realizations of random variables (in contrast to the random variables themselves). For example,  $p_{\mathbf{x}}(\boldsymbol{\xi})$ ,  $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{\xi} | \boldsymbol{\nu})$ ,  $\mathbb{E}[\mathbf{x} | \mathbf{y} = \boldsymbol{\nu}]$ , etc.
- The set of probability distributions on a finite set  $\mathcal{X}$  is denoted  $\Delta(\mathcal{X})$ . (For  $\mathcal{X} = [n]$ , this is the probability simplex which can be embedded into  $\mathbb{R}^n$ ).
- The Gaussian distribution with mean  $\boldsymbol{\mu}$  and covariance  $\boldsymbol{\Sigma}$  is denoted  $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ .
- The uniform distribution over a compact set  $\mathcal{X}$  is denoted  $\mathcal{U}(\mathcal{X})$ .

## Machine Learning Notation

- Encoders and denoisers are usually denoted by  $f$  and usually have parameters  $\theta \in \Theta$ . We usually write the features  $\mathbf{z} = f_{\theta}(\mathbf{x})$ .
- Decoders are usually denoted by  $g$  and usually have parameters  $\eta$ . We usually write the auto-encoding  $\hat{\mathbf{x}} = g_{\eta}(\mathbf{z})$  corresponding to  $\mathbf{x}$ .
- When  $f$  and  $g$  are implemented as neural networks, they will have the same number of layers without loss of generality; we write them as  $f = f^L \circ f^{L-1} \circ \dots \circ f^2 \circ f^1 \circ f^{\text{pre}}$  and  $g = g^{\text{post}} \circ g^L \circ g^{L-1} \circ \dots \circ g^2 \circ g^1$ .
- We write the features at the input to layer  $\ell$  as  $\mathbf{z}^{\ell}$  such that  $\mathbf{z}^{\ell+1} = f^{\ell}(\mathbf{z}^{\ell})$  with  $\mathbf{z}^1 = f^{\text{pre}}(\mathbf{x})$  and  $\mathbf{z} = \mathbf{z}^{L+1}$ . Similarly the autoencoding features are  $\hat{\mathbf{x}}^{\ell}$  with  $\hat{\mathbf{x}}^{\ell+1} = g^{\ell}(\hat{\mathbf{x}}^{\ell})$  with  $\hat{\mathbf{x}}^1 = \mathbf{z}$  and  $\hat{\mathbf{x}} = g^{\text{post}}(\hat{\mathbf{x}}^{L+1})$ .
- For denoising, optimization, and processes which have a continuous time index, the time is almost always a subscript, e.g.,  $\mathbf{x}_t$ . Discrete sequences can have the index be a superscript or subscript.

## Modeling and Optimization Notation

- As in the Probability and Machine Learning Notation sections above, when we have a model, say for the realizations of a data distribution supported on  $\mathbb{R}^D$ , we always denote it with “plain” accented variables, say  $\mathbf{x}$ .
- In cases where we assume our data  $\mathbf{x}$  is generated by a model from a restricted class of *parametric models*, say with a parameter  $\boldsymbol{\mu}$ , we also denote the *true parameter* with plain accenting.<sup>3</sup>

<sup>3</sup>This could arise either due to us making an assumption about our data, or due to our data being synthetically generated. This convention allows us to adopt a concise notation across both ‘probabilistic’ and ‘analytical’ modeling setups. Readers more familiar with the latter kinds of frameworks may be used to using notation such as  $\boldsymbol{\mu}_{\#}$  or  $\boldsymbol{\mu}_o$  for this case; contrast with our accented decision variable notation.

- *Decision variables* (or learnable parameters) of a model fit to observed data are denoted with “tilde” accenting, e.g.  $\tilde{\mathbf{x}}$ , in cases where there is an underlying model  $\mathbf{x}$ , or with plain notation in cases where no confusion is possible (e.g., if an empirically-fit model for *nonparametric* data  $\mathbf{x}$  involves “mean” parameters  $\boldsymbol{\mu}$ ).
- *Optimal solutions* to optimization problems are denoted with “star” accenting, either as a superscript or a subscript (say  $\mathbf{x}_*$  or  $\mathbf{x}^*$ ).
- *Estimators* for data that correspond to a specific statistical model, especially for the mean of that statistical model, are denoted with “bar” accenting, say  $\bar{\mathbf{x}}$  (especially for minimum mean-squared error denoising in Chapter 3).
- *Approximations* associated to computational procedures for modeling data are denoted with “hat” accenting, say  $\hat{\mathbf{x}}$  for the output of an autoencoder or a generative model that approximates data  $\mathbf{x}$ .<sup>4</sup>

---

<sup>4</sup>We generally distinguish these cases from optimal solutions  $\mathbf{x}_*$ , although in some special cases they are equal (e.g., in several examples in Chapter 2). Think of the distinction between the parameter  $\boldsymbol{\mu}$  of a model, which might be fit with optimization, and the results of sampling with that learned parameter.

# Contents

<b>Preface</b>	<b>ii</b>
<b>Preface to Version 2.0</b>	<b>vi</b>
<b>Declaration of Open Source</b>	<b>ix</b>
<b>Acknowledgment</b>	<b>xi</b>
<b>Notation</b>	<b>xiii</b>
<b>1 An Informal Introduction to Intelligence</b>	<b>1</b>
1.1 Intelligence, Cybernetics, and Artificial Intelligence . . . . .	1
1.2 What to Learn? . . . . .	9
1.2.1 Predictability . . . . .	9
1.2.2 Low Dimensionality . . . . .	11
1.3 How to Learn? . . . . .	17
1.3.1 Analytical Model-Based Approaches . . . . .	17
1.3.2 Empirical Data-Driven Approaches . . . . .	26
1.4 A Unifying Approach . . . . .	36
1.4.1 Learning Parsimonious Representations . . . . .	37
1.4.2 Learning Informative Representations . . . . .	41
1.4.3 Learning Consistent Representations . . . . .	42
1.4.4 Learning Self-Consistent Representations . . . . .	45
1.5 Bridging Theory and Practice for Machine Intelligence . . . . .	46
<b>2 Learning Linear and Independent Structures</b>	<b>51</b>
2.1 A Low-Dimensional Subspace . . . . .	53
2.1.1 Principal Components Analysis (PCA) . . . . .	53
2.1.2 Pursuing Low-Rank Structure via Power Iteration . . . . .	58
2.1.3 Local and Incremental Learning via Online PCA . . . . .	59
2.1.4 The Statistical View: Probabilistic PCA . . . . .	60
2.1.5 Masked Low-Rank Matrix Completion . . . . .	62
2.2 A Mixture of Complete Low-Dimensional Subspaces . . . . .	64
2.2.1 Mixtures of Subspaces and Sparsely-Used Dictionaries . . . . .	65
2.2.2 Complete Dictionary Learning . . . . .	67

---

2.2.3	Connection to ICA and Kurtosis	70
2.3	A Mixture of Overcomplete Low-Dimensional Subspaces	73
2.3.1	Sparse Coding with an Overcomplete Dictionary	74
2.3.2	Overcomplete Dictionary Learning	76
2.3.3	Learned Deep Sparse Coding	78
2.4	Summary and Notes	81
2.5	Exercises and Extensions	83
<b>3</b>	<b>Pursuing Low-Dimensional Distributions via Denoising</b>	<b>91</b>
3.1	Entropy Minimization and Compression	93
3.1.1	Entropy and Coding Rate	93
3.1.2	Differential Entropy	93
3.1.3	Minimizing Coding Rate	94
3.2	Compression via Denoising	96
3.2.1	Diffusion and Denoising Processes	96
3.2.2	Sampling a Distribution via Iterative Denoising	109
3.3	Memorization, Generalization, and Coding Rates	118
3.4	Summary and Notes	124
3.5	Exercises and Extensions	125
<b>4</b>	<b>Representation Learning via Lossy Compression</b>	<b>129</b>
4.1	Compression via Lossy Coding	131
4.1.1	Necessity of Lossy Coding	131
4.1.2	Rate Distortion and Data Geometry	133
4.1.3	Lossy Coding Rate for a Low-Dimensional Gaussian	139
4.1.4	Clustering a Mixture of Low-Dimensional Gaussians	142
4.2	Transformation via Maximizing Information Gain	150
4.2.1	Representation Learning from Classified Data	151
4.2.2	Linear Discriminative Representations	154
4.2.3	The Principle of Maximal Coding Rate Reduction	157
4.2.4	Optimization Properties of Coding Rate Reduction	162
4.3	Learning Representations for Imagery Data	165
4.3.1	Supervised Representation Learning	165
4.3.2	Unsupervised Representation Learning	166
4.4	Summary and Notes	171
4.5	Exercises and Extensions	172
<b>5</b>	<b>Deep Representations as Unrolled Optimization</b>	<b>175</b>
5.1	White-Box Deep Networks via Unrolled Optimization	177
5.1.1	Deep Networks from Unrolled Gradient Descent	178
5.1.2	Convolutional Networks from Invariant Rate Reduction	184
5.2	White-Box Transformers from Unrolled Optimization	194
5.2.1	Unrolled Optimization for Sparse Rate Reduction	195
5.2.2	Overall White-Box Transformer Architecture: CRATE	201
5.3	Variants of Deep Architectures by Design	204
5.3.1	Attention-Only Transformer Architecture	205

---

5.3.2	Linear-Time Attention: Token Statistics Transformer . . .	208
5.3.3	Causal CRATE . . . . .	213
5.4	Summary and Notes . . . . .	218
5.5	Exercises and Extensions . . . . .	219
<b>6</b>	<b>Consistent and Self-Consistent Representations</b>	<b>223</b>
6.1	Learning Consistent Representations . . . . .	225
6.1.1	Linear Autoencoding via PCA . . . . .	227
6.1.2	Nonlinear PCA and Autoencoding . . . . .	228
6.1.3	Sparse Autoencoding . . . . .	230
6.1.4	Variational Autoencoding . . . . .	231
6.1.5	Joint Representation and Distribution Learning . . . . .	234
6.2	Learning Self-Consistent Representations . . . . .	236
6.2.1	Closed-Loop Transcription via Stackelberg Games . . . . .	239
6.2.2	A Mixture of Low-Dimensional Gaussians . . . . .	244
6.3	Continuous Learning Self-Consistent Representations . . . . .	247
6.3.1	Class-wise Incremental Learning . . . . .	247
6.3.2	Sample-wise Continuous Unsupervised Learning . . . . .	252
6.4	Summary and Notes . . . . .	256
6.5	Exercises and Extensions . . . . .	258
<b>7</b>	<b>Inference with Low-Dimensional Distributions</b>	<b>259</b>
7.1	Bayesian Inference and Constrained Optimization . . . . .	260
7.1.1	Bayesian Inference with Low-Dimensional Distributions . . . . .	260
7.1.2	Constrained Optimization with Submanifolds . . . . .	262
7.1.3	Representative Practical Settings for Inference . . . . .	265
7.2	Conditional Inference with a Known Data Distribution . . . . .	266
7.3	Conditional Inference with a Learned Data Representation . . . . .	269
7.3.1	Image Completion with Masked Auto-Encoding . . . . .	269
7.3.2	Conditional Sampling with Measurement Matching . . . . .	270
7.4	Conditional Inference with Paired Data and Measurements . . . . .	283
7.4.1	Class-Conditioned Image Generation . . . . .	285
7.4.2	Caption-Conditioned Image Generation . . . . .	296
7.5	Conditional Inference with Measurement Self-Consistency . . . . .	302
7.5.1	Linear Measurement Models . . . . .	303
7.5.2	Learning 3D World Model from 2D Images . . . . .	303
7.6	Summary and Notes . . . . .	308
7.7	Exercises and Extensions . . . . .	309
<b>8</b>	<b>Learning Representations for Real-World Data and Tasks</b>	<b>311</b>
8.1	Introduction . . . . .	311
8.1.1	Outline of the Chapter . . . . .	312
8.1.2	Overall Technical Setup . . . . .	313
8.2	Unsupervised Learning of Image Representations . . . . .	315
8.2.1	Data . . . . .	315
8.2.2	The Simplified DINO Task and Objective . . . . .	317

---

8.2.3	Architecture: Vision Transformer . . . . .	320
8.2.4	Optimization Strategy . . . . .	324
8.2.5	Evaluation Methodology . . . . .	328
8.2.6	Experimental Setup and Results . . . . .	330
8.3	Weakly Supervised Image Representation via Text Binding . . . . .	333
8.3.1	Data . . . . .	333
8.3.2	The CLIP Task and Objective . . . . .	334
8.3.3	Architecture: Vision Tower . . . . .	335
8.3.4	Architecture: Text Tower . . . . .	335
8.3.5	Optimization Strategy . . . . .	336
8.3.6	Evaluation Methodology . . . . .	337
8.3.7	Experimental Setup and Results . . . . .	338
8.3.8	Simplified Extension: LIFT . . . . .	340
8.4	Supervised Image Representation via Classification . . . . .	341
8.4.1	Task and Objective . . . . .	342
8.4.2	The CRATE Architecture . . . . .	342
8.4.3	Optimization . . . . .	343
8.4.4	Evaluation Methodology . . . . .	344
8.4.5	Experimental Setup and Results . . . . .	344
8.5	Image Representation via Masked Autoencoding . . . . .	346
8.5.1	The MAE Task and Objective . . . . .	347
8.5.2	CRATE-Based MAE Architecture . . . . .	347
8.5.3	Optimization . . . . .	349
8.5.4	Evaluation . . . . .	349
8.5.5	Experiments . . . . .	350
8.6	Image Generation via Auto-Encoding and Sampling . . . . .	352
8.6.1	Variational Auto-Encoding of Imagery Data . . . . .	352
8.6.2	Representation Auto-Encoding of Imagery Data . . . . .	354
8.6.3	Sampling from Learned Representations via Denoising . . . . .	357
8.6.4	Realizing Denoising with a U-Net . . . . .	358
8.6.5	Realizing Denoising with a Diffusion Transformer . . . . .	361
8.7	Conditioned Image Representation and Generation . . . . .	365
8.7.1	Task Formulation . . . . .	365
8.7.2	The Basic Realization: Text-to-Image Generation . . . . .	366
8.7.3	Advanced Conditioning Mechanisms . . . . .	370
8.7.4	Extension to Video Generation . . . . .	374
8.8	Image and Text Conditioned 3D Object Generation . . . . .	376
8.8.1	3D Shape Representations, Datasets, and Tokenization . . . . .	376
8.8.2	Task . . . . .	381
8.8.3	Architecture and Objective . . . . .	382
8.8.4	Experiments and Analysis . . . . .	385
8.9	Generation-Based 3D Shape and Pose Reconstruction . . . . .	391
8.9.1	Task and Objective . . . . .	392
8.9.2	3D Shape Representation and Tokenization . . . . .	394
8.9.3	Architecture and Implementation . . . . .	395
8.9.4	Cascaded Flow Modeling . . . . .	396

---

8.9.5	Experiments	398
8.9.6	Conclusion	404
8.10	Conditioned Human Body Motion Generation	405
8.10.1	Representing Human Bodies	405
8.10.2	Task and Setup	407
8.10.3	Designing the Conditioning Representation	408
8.10.4	The Conditional Prior	409
8.10.5	Incorporating Measurements via Guidance	411
8.10.6	Conditional Sampling Algorithm	412
8.10.7	Insights from Experiments	413
8.10.8	Discussion	416
8.11	Natural Language Representation and Generation	418
8.11.1	Data	418
8.11.2	Task and Objective	419
8.11.3	Architecture: Causal CRATE	422
8.11.4	Optimization Strategy	424
8.11.5	Evaluation Methodology	425
8.11.6	Experimental Setup and Results	425
8.12	Scaling and Improving White-Box Transformers	427
8.12.1	Increasing Network Width: CRATE- $\alpha$	427
8.12.2	Linear Time Complexity Transformers	430
8.12.3	Attention-Only Transformers	431
8.13	Summary and Notes	433
8.14	Exercises and Extensions	434
<b>9</b>	<b>Open Problems and Directions about Intelligence</b>	<b>437</b>
9.1	Towards Autonomous Intelligence: Close the Loop?	438
9.2	Towards Natural Intelligence: Beyond Back Propagation?	439
9.3	Towards Scientific Intelligence: Beyond the Turing Test?	442
9.3.1	The Evolution of Intelligence in Nature	442
9.3.2	From Inductive to Deductive Intelligence	443
9.3.3	Scientific Tests of Intelligence	445
	<b>Appendices</b>	<b>449</b>
<b>A</b>	<b>Optimization Methods</b>	<b>449</b>
A.1	Steepest Descent	449
A.1.1	Vanilla Gradient Descent for Smooth Problems	450
A.1.2	Preconditioned Gradient Descent for Badly-Conditioned Problems	455
A.1.3	Proximal Gradient Descent for Non-Smooth Problems	457
A.1.4	Stochastic Gradient Descent for Large-Scale Problems	459
A.1.5	Putting Everything Together: Adam	460
A.2	Computing Gradients via Automatic Differentiation	462
A.2.1	Differentials	463

---

A.2.2	Automatic Differentiation . . . . .	466
A.2.3	Back Propagation . . . . .	467
A.3	Game Theory and Minimax Optimization . . . . .	471
A.3.1	Learning Stackelberg Equilibria . . . . .	474
A.3.2	Practical Considerations when Learning Stackelberg Equilibria . . . . .	478
A.4	Exercises . . . . .	478
<b>B</b>	<b>Entropy, Diffusion, Denoising, and Lossy Coding</b>	<b>481</b>
B.1	Differential Entropy of Low-Dimensional Distributions . . . . .	482
B.2	Diffusion and Denoising Processes . . . . .	483
B.2.1	Diffusion Process Increases Entropy Over Time . . . . .	484
B.2.2	Denoising Process Reduces Entropy Over Time . . . . .	486
B.2.3	Technical Lemmas and Intermediate Results . . . . .	490
B.3	Lossy Coding and Sphere Packing . . . . .	500
B.3.1	Proof of Relationship Between Rate Distortion and Covering . . . . .	501
B.3.2	Proof of Lemma B.6 . . . . .	504
	<b>Bibliography</b>	<b>507</b>

