Towards Open Ended Learning: Budgets, Model Selection, and Representation

Thesis by

Ryan Geoffrey Gomes

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2011

(Defended January 18, 2011)

For my family.

Acknowledgements

My time at Caltech has been like no other in my life. I am grateful for having had the opportunity to immerse myself in such an intellectually stimulating environment, all while enjoying the beautiful Southern Californian weather and lifestyle. As I finish this thesis I have the pleasure of recounting the people that have shaped my experience; indeed, they have shaped and sharpened the very manner in which I think.

I'd first like to thank my thesis adviser Prof. Pietro Perona. Pietro is a great source of wisdom and ideas, and I have had tremendous fun engaging in brainstorming and research sessions with him. I was always amazed at his ability to understand me, even when I didn't understand my ideas well enough myself to express them properly. Indeed, Pietro has taught me a great deal about how best to communicate complex research. Because of his guidance, the Vision Lab at Caltech is a wonderful place to work. I have had the opportunity to meet and interact with many highly distinguished researchers in the fields of computer vision and machine learning, who make it a point to visit his lab.

I have had the privilege of collaborating with two other professors, from whom I have learned a great deal. Max Welling has an infectious enthusiasm for research and was a great source of encouragement. I benefited significantly from working with Max, as well as reading his papers and technical notes. It has also been a pleasure working with Andreas Krause, a talented young professor of machine learning. On many occasions his insight has provided me with the right perspective to solve challenging issues, and I admire his focus on attacking practical problems while maintaining theoretical rigour.

My fellow members of the Vision Lab have enriched my time at Caltech as intellectual peers and as friends. Merrielle and I shared many a wry laugh as we navigated the currents of graduate school, both of us having entered the Computation and Neural Systems program and the Vision Lab at the same time. I enjoyed many interesting conversations with Piotr and Peter over the time we shared as labmates, and they have very much shaped my ideas on computer vision. Marco's sense of humor and helpful advice have been much appreciated as well. To the rest of the members of the lab, past and present, thank you!

I'd like to thank my great group of friends, many of whom I've known since high school or college, for keeping me sane and providing welcome respite from the challenges of pursuing my Ph.D. Finally, I wish to thank my wonderful family. My father and mother have encouraged and supported me every step of the way, and they have provided me with the foundation and freedom to follow my interests. My brother and sister have also been a great source of encouragement, and I very much love being their big brother.

Abstract

Biological organisms learn to recognize visual categories continuously over the course of their lifetimes. This impressive capability allows them to adapt to new circumstances as they arise, and to flexibly incorporate new object categories as they are discovered. Inspired by this capability, we seek to create artificial recognition systems that can learn in a similar fashion.

We identify a number of characteristics that define this *Open Ended* learning capability. Open Ended learning is *unsupervised*: object instances need not be explicitly labeled with a category indicator during training. Learning occurs *incrementally* as experience ensues; there is no training period that is distinct from operation and the categorization system must operate and update itself in a timely fashion with limited computational resources. Open Ended learning systems must flexibly adapt the number of categories as new evidence is uncovered.

Having identified these requirements, we develop Open Ended categorization systems based on probabilistic graphical models and study their properties. From the perspective of building practical systems, the most challenging requirement of Open Ended learning is that it must be carried out in an unsupervised fashion. We then study the question of how best to represent data items and categories in unsupervised learning algorithms in order to extend their domain of application.

Finally, we conclude that continuously learning categorization systems are likely to require human intervention and supervision for some time to come, which suggests research in how best to structure machine-human interactions. We end this thesis by studying a system that reverses the typical role of human and machine in most learning systems. In *Crowd Clustering*, humans perform the fundamental image

categorization tasks, and the machine learning system evaluates and aggregates the results of human workers.

Contents

Chapter 1

Introduction

1.1 Object Categorization and Machine Learning

Visual object categorization is a problem of central importance in vision science. Behaving organisms categorize their visual world in order to survive. As an engineering goal, automatic visual object categorization would enable an untold number of applications, and would change the world in ways that we can scarcely imagine today.

While humans can effortlessly categorize objects, it has proven fiendishly difficult to build artificial systems with the same capabilities. Recently, the computer vision community has identified Machine Learning as a promising route towards automated recognition systems. Rather than relying on human expertise to explicitly codify category definitions, the machine learning approach uses training examples to guide the selection of a classifier from a large parameterized set of alternative classifiers; the hope being that the selected classifier will correctly perform the categorization task not only on the training images but also on novel object instances.

The application of machine learning to object categorization has led to practical success in constrained settings. For example, the problem of detecting faces can be cast as a constrained categorization task: classify an image window into the "face" category if it contains a face, or the "not face" category if it does not. The Viola and Jones face detector [VJ01] is an impressive milestone of computer vision and forms the basis of a number of commercial systems. There is recent promising work in object detection of more complex articulated categories such as pedestrians, e.g.,

Felzenszwalb et al. [FMR08]. There is still quite a ways to go [DWSP09].

Numerous challenges remain in extending this success to systems capable of handling multiple categories. Currently, systems capable of detecting objects in cluttered natural scenes require a large number of closely cropped training image windows. The substantial supervisory effort required to collect such a data set is a barrier towards building systems that can handle a large number of object categories. Ongoing work has focused on reducing the required amount of supervision. For example, Weber et al. [WWP00] introduce the notion of weak supervision: a training image need only have an instance of the category to be learned, the effort to precisely locate it in the image is not necessary. An additional challenge in constructing multi-category recognition systems is that each object category is unique and is best identified by a set of properties specific to that category. The work of Varma and Ray [VR07] addresses this by learning category-specific weightings of multiple image kernels.

It is clear that artificial systems lag behind biological organisms in terms of categorization accuracy and in the number of recognized classes. In addition, biological systems display a flexibility of adaptation and learning that is not present in our engineered solutions. Most artificial object categorization systems involve learning according to a standard paradigm which we define here. First, a *batch* training set is collected that has instances of every category that the system needs to learn. Training is typically *supervised* in that an indication of each training image's category is available (see [GD06] for an exception.) Once the recognition system is trained on this batch data set, it is fixed. During operation of the system (known as the *test* phase), the system estimates the category membership of test images, but does not learn anything new from them. The system is fixed and can not adapt. Existing categories can not be refined, nor are new categories added as they are encountered.

1.2 Open Ended Learning

In contrast to this standard paradigm, biological systems do not make a distinction between a training and testing phase and do not require a batch training set. An

organism would be at a significant disadvantage if it had to be exposed to training examples from every category before it could recognize objects! Instead, learning is *incremental*: visual information arrives continuously, and categorization and learning happen simultaneously. New visual information may lead to refinement of existing categories. For example, we may learn over time that the dog category includes poodles. New categories may be added: a mammal is sure to recognize a new type of predator in the future after a close call. Perhaps most significantly, biological organisms appear to require little in the way of explicit supervision to recognize novel objects.

Such adaptive learning would be very useful in a number of engineering applications as well. Consider a surveillance camera that surveys a scene. Over time, new objects could be introduced and the system would first recognize them as novel (perhaps triggering an alarm.) These novel object categories would then be incorporated into the collection of known categories. This might be generalized to learning categories of object activities and behaviors as well. It would be beneficial to have much of this learning happen in an unsupervised fashion, rather than requiring operator effort to provide feedback on all visual information or detected objects.

The Internet may be viewed as a constantly evolving collection of images. A "webcrawler" software agent could explore the web, examining and categorizing images and flexibly adding and revising categories as necessary. This type of exploratory system might be part of a larger system of organizing visual information into a type of visual encyclopedia such as that proposed by Perona [Per10].

In support of such applications, we define the notion of *Open Ended* learning systems, which have the following set of characteristics:

- **Incremental learning.** There is no distinction between training and test phases; learning and category prediction are performed simultaneously. Open Ended learning systems may process data instances one at a time or in small batches. This is also sometimes referred to as *online* learning.

- **Computational Resource Budgets.** Incremental learning should occur un-

der the constraint of computational resource *budgets*. Naively, an incremental learning system could be constructed from a batch learner by storing in memory every training instance ever encountered. The batch learning process would be run from scratch each time new data is added to the collection. This is undesirable because the memory required to store examples would grow without bound over the life span of the system. The amount of time required to update the system with new evidence would also increase (perhaps dramatically) over the life span of the system. Therefore, we impose the notion of limited memory or time budgets that the learning system may not exceed when updating the system with new evidence. Biological systems must also operate under analogous finite resource budgets.

- **Unsupervised learning.** Open Ended systems must be able to handle unlabeled data instances that are not provided with a ground truth value indicating the item's true category membership. Optionally, an Open Ended system may be operated with partial labels, which is known as *semi-supervised* learning in the literature [CSZ06]. However, in contrast with existing semi-supervised learning approaches, an Open Ended system must be able to handle latent categories that do not have a single labeled instance.

- **Model Selection.** An Open Ended learning system must be able to flexibly update the number of categories as evidence arrives. Categories may be created or destroyed as determined by the structure of the data itself. This is often referred to as *Model Selection* in the statistical literature [CB] [Zuc00].

The performance of an Open Ended learning system may be quantitatively measured according to methods used in standard unsupervised learning. This involves measuring the system's performance on *held out* data that is not used to update the learning system. Performance measures may be based on the system's ability to predict the held out data; examples of predictive measures include data likelihood (see Section 2.6) or quantization performance relative to a distance measure (see Section 4.7). Alternatively, we need not make use of held out data. If ground truth

category labels are available for our data, then we may compare the unsupervised categorization produced by the learning system to the true category memberships of the training data. A number of measures exist for this purpose (see Section 5.7.1.)

The astute reader may notice an apparent contradiction in our definition of Open Ended learning. We desire learning systems that may add new categories as they are discovered during operation, yet we require that our systems operate under finite resource budgets. Won't the resource budget eventually place a limit on the complexity of the categorization system that may be learned, in effect limiting its ability to add new categories indefinitely? Indeed, this is likely to be the case. However, in practice we are concerned with systems that may add categories and grow in complexity for as long as possible, given their finite resource limits. We show that it is possible to build learning systems that manage this tradeoff much more effectively than existing work.

1.2.1 Related Work

There have been proposals in the literature that are related to our notion of Open Ended learning. Thrun and Mitchell [TM95] define *Lifelong Learning* as an approach to robotic control problems. Lifelong Learning is concerned with transferring experience learned from one robotic control task to another, in an attempt to reduce the difficulty of learning the subsequent task. We agree that transfer learning is likely of central importance in terms of incrementally learning collections of large object categories. Ideally, the large number of training examples necessary for learning the first categories can be leveraged to reduce the effort associated with learning later categories. See [OPZ06] for an account in the vision literature. However, Lifelong Learning differs from Open Ended learning in that it is concerned with reinforcement learning problems in which performance feedback is available as an implicit supervisory signal. In contrast, we are concerned with unsupervised learning of categorization systems.

Within the visual recognition literature, the Ph.D thesis of Justus Piater [Pia01]

makes similar observations about the closed nature of the standard machine learning paradigm as applied to visual recognition, and they advocate a continuous learning approach. While their system has no fixed limit on the number of object categories (they learn a naive Bayes classifier for each category), they operate in the supervised learning framework. Their contribution is an online learning system for supervised visual feature selection and classifier training.

Carlson et al.'s Never Ending Language Learning (NELL) system [CBK+10] is very recent work by Mitchell's group that meets much of our definition of Open Ended learning. The system takes language examples from the Web, and uses them to learn a growing knowledge base of noun phrase categories, as well as logical relations such as "X plays for Y" or "X is mayor of Y" between noun phrase categories. The system is capable of proposing and accepting new categories, which may be interpreted as being an instance of model selection.

While the system bootstraps itself from some labeled examples, it is able to handle induction of categories without requiring labeled instances of the new category. The system does make use of 10 to 15 minutes of human interaction per day, which are aimed at pruning out mistaken categories induced by the system (see the Postscript for a discussion of human interaction in Open Ended learning.) NELL learns from a fixed corpus of text, so it is unclear how the system scales computationally as the corpus size increases. NELL is based in part on years of specialized research in language concept induction. This thesis is aimed at developing Open Ended learning systems for general purpose pattern categorization.

1.3 Thesis Organization

This thesis is organized as a series of chapters (2, 3, 4 & 5) that are adapted from self-contained articles that were previously published in other formats. We end with a Postscript that includes our conclusions about the prospects for Open Ended learning and future research directions, as well as preliminary research along a direction that may be interpreted as a counterpoint to the core of this thesis.

	Ch. 2 & Ch. 3	Ch. 4	Ch. 5
unsupervised	✓	✓	✓
incremental with budget	✓	✓	
model selection	✓		✓
data representation	Euclidean points	Arbitrary objects with "distance"	Arbitrary objects with PSD kernels
category representation	Gaussian Distribution	Prototype Example	Discriminative Classifier

Table 1.1: Thesis organization: Chapters 2, 3, 4, and 5 are self contained articles that investigate aspects of Open Ended learning as well as issues of data and category representation in unsupervised learning.

Our general approach towards realizing Open Ended learning is to begin from a foundation based on unsupervised learning methods, and then to extend them into Open Ended learning algorithms. Chapter 2 involves an Open Ended variant of the statistical mixture model, which is a simple model for categorizing data items that lie in a Euclidean feature space. Chapter 3 details an Open Ended variant of the Topic Model, which is an hierarchical extension of the mixture model that is suitable for handling images and documents which may be collections of more than one category. Our algorithms continuously adapt and expand their collection of categories as data arrives, while maintaining fixed computational budgets. We show that our approach outperforms other online learning algorithms, and performs nearly as well as batch algorithms that have complete access to the entire data history. Both chapters are joint work with Max Welling and Pietro Perona.

Because unsupervised learning forms the bedrock of our approach, we then study the extent to which data and category representation in unsupervised learning can impact the performance of our learning systems. Chapters 4 and 5 represent work in which we relax some of the requirements of Open Ended learning for the sake of studying alternate representations that are more flexible than the Gaussian distributed categories and Euclidean feature spaces of Chapter 2 and Chapter 3.

Chapter 4 (joint work with Andreas Krause) develops a general framework for selecting data examples from streaming data sources, while maintaining computational budgets. These selected instances may be used as prototype examples in nonparamet-

ric learning algorithms, and we develop algorithms for unsupervised categorization and supervised regression problems. In the case of unsupervised categorization, data items may have any structure that has a "pseudo-distance" defined over pairs. This pseudo-distance need not be a metric, and so in principle our method can handle complex representations suitable for difficult application domains. We prove theoretical performance guarantees for our algorithms, and demonstrate their efficacy on large scale problems.

Chapter 5 (with Andreas Krause and Pietro Perona) further explores data and category representation in unsupervised learning. We develop a probabilistic technique for training discriminative multi-category classifiers without supervisory labels. Rather than making constructive or generative definitions of categories, we model the decision boundaries between them. This results in a rich and flexible category representation. The framework is also capable of handling data items with arbitrary structure, provided they have a positive semi-definite kernel function defined between pairs of items. We show that the resulting algorithm outperforms other approaches on numerous real world datasets.

The inter-relationships between the core chapters are summarized in Table 1.1. Part III contains the Postscript, with concluding thoughts about the limitations of current approaches to unsupervised learning in real world applications. We then suggest research directions that might lead past these limitations by exploring Open Ended learning systems which involve efficient use of human expertise. The final chapter (with Peter Welinder, Andreas Krause, and Pietro Perona) presents a system in which human expertise is treated as a foundational computational block for categorization rather than as a scarce resource. The system may be used to learn a representation of images while categories emerge naturally without having to be pre-defined.

Part I

Incremental Unsupervised Learning with Budgets

Chapter 2

Incremental Learning of Nonparametric Bayesian Mixture Models

2.1 Abstract

Clustering is a fundamental task in many vision applications. To date, most clustering algorithms work in a batch setting and training examples must be gathered in a large group before learning can begin. Here we explore incremental clustering, in which data can arrive continuously. We present a novel incremental model-based clustering algorithm based on nonparametric Bayesian methods, which we call Memory Bounded Variational Dirichlet Process (MB-VDP). The number of clusters are determined flexibly by the data and the approach can be used to automatically discover object categories. The computational requirements required to produce model updates are bounded and do not grow with the amount of data processed. The technique is well suited to very large datasets, and we show that our approach outperforms existing online alternatives for learning nonparametric Bayesian mixture models.

2.2 Introduction

Discovering visual categories automatically with minimal human supervision is perhaps the most exciting current challenge in machine vision [WWP00, SRE+05]. A

related problem is quantizing the visual appearance of image patches, e.g., to build dictionaries of visual words in order to train recognition models for textures, objects, and scenes [LM99, VNU03, DS03, FFP05, JT05]. This second problem is easier, because the features (e.g., pixels, SIFT coordinates) have been agreed upon in advance and do not need to be discovered as part of the process. In both cases unsupervised clustering is an important building block of the system.

Unsupervised clustering is usually carried out in batch on the entire training set. Here we consider instead 'incremental' or 'on line' unsupervised clustering. There are two reasons why incremental clustering or category learning may be useful. First of all, an organism, or a machine, has a competitive advantage if it can immediately use all the training data collected so far— rather than wait for a complete training set. Second, incremental methods usually have smaller memory requirements: new training examples are used to update a 'state' and then the examples are forgotten. The state summarizes the information collected so far – it typically consists of a parametric model and it thus occupies a much smaller amount of memory than a full-fledged training set. So: when the system has to operate while learning, when the memory available is small (as in an embedded system), or when the training data are very voluminous, an incremental method is the way to go. It has to be expected that an on-line method is not as efficient in extracting information from the data as a batch method. This is because decisions must often be taken without the benefit of future information.

A challenge for clustering methods, one that is often swept under the rug, is determining the complexity of the final model: "How many clusters should I plan for?" Batch methods have the luxury of solving this question by trial-and-error: fit many models, from simple to complex, and pick the one that maximizes some criterion, e.g., the likelihood on a validation set. Estimating the complexity of the model is much harder for on-line methods. Furthermore, the complexity is likely to grow with time, as more training examples are acquired and stronger evidence is available for subtler distinctions.

We present a new approach for learning nonparametric Bayesian mixture models

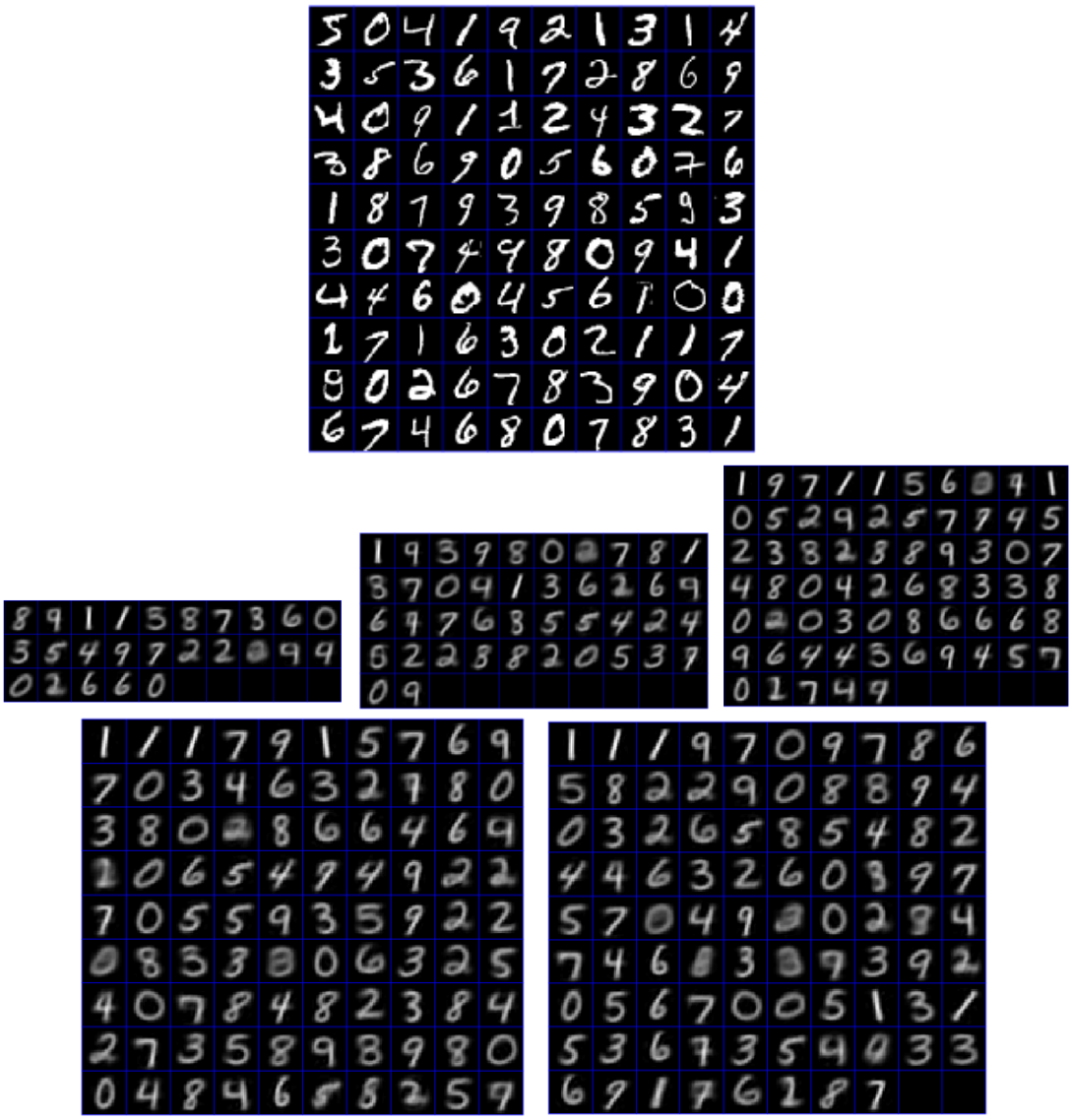


Figure 2.1: Top: A sample of the inputs to the incremental learning process. Middle: Cluster means discovered by incremental algorithm after 6000, 12000, and 30000 digits processed. As expected, the model complexity increases as data arrives. The computational burden per model update is not a function of the number of data points processed; it grows more slowly with the number of clusters discovered. Bottom Left: Cluster centers produced by incremental algorithm after visiting all 60000 digits, with effective memory size of 6000 digits. Bottom Right: Cluster centers produced by batch algorithm. Clusters are ordered according to size, from top left to bottom right. Our incremental algorithm requires substantially less memory and is faster than the comparable batch algorithm. See Section 2.5 for a description of the algorithm and Section 2.6 for more information about the experimental results.

incrementally. Our approach has a number of desirable properties: it is incremental, it is non-parametric in the number of components of the mixture, its memory use is parsimonious and bounded. Empirically, we find that it makes good use of the information provided by the training set, almost as good as a batch method, while being faster and able to tackle problems the size of which a batch method is unable to approach.

Section 2.3 provides background on the Dirichlet Process mixture model and sufficient statistics. Section 2.4 briefly describes existing approaches to the problem and Section 2.5 explains our approach. Section 2.6 shows experimental results on an object recognition problem, clustering of MNIST digits, and a large image patch clustering experiment. Discussions and conclusions may be found in Section 2.7.

2.3 Background

We start by reviewing the Dirichlet Process mixture model (DPMM) [Ant74].

2.3.1 Dirichlet Process Mixture Model

The DPMM extends the traditional mixture model to have an infinite number of components. Data points x_t are assumed to be drawn i.i.d. from the distribution:

$$p(x_t) = \sum_{k=1}^{\infty} \pi_k p(x_t|\phi_k), \tag{2.1}$$

where ϕ_k are component parameter vectors and π_k are a set of mixing weights that sum to 1. During inference, the mixing weights and the component parameter vectors are treated as random quantities. A probabilistic structure known as the Dirichlet Process [Fer73] defines a prior on these random variables.

The component parameters ϕ_k are assumed to be independent samples from a probability distribution H. The mixture weights π_k may be constructed from a count-

ably infinite set of *stick breaking* random variables V_k [Set94] according to

$$\pi_k = V_k \prod_{i=1}^{k-1} (1 - V_i). \tag{2.2}$$

The stick breaking variables are distributed independently according to $V_k \sim Beta(1, \alpha)$, where $\alpha > 0$ is the concentration parameter of the Dirichlet Process. When α is small, there is a bias towards a small number of large mixing weights (clusters), and when it is large there is a tendency to have many small weights. The mixing weights are guaranteed to sum to one, as required to make a well-defined mixture model.

It is convenient to introduce a set of auxiliary assignment variables $Z = \{z_1, \ldots, z_N\}$, one for each data point x_t. $z_t \in \mathbb{N}$ designates the mixture component that generated data point x_t. The assignment variables Z specify a clustering or partition of the data.

In learning, we are interested in estimating the posterior $p(Z, \Phi, V|X, \alpha, H)$ given a set of observations $X = \{x_1, \ldots, x_N\}$. We assume that the component parameter prior H and concentration parameter α are known.

2.3.2 Exponential Family and Sufficient Statistics

We will restrict ourselves to component distributions that are members of the exponential family [BS94], because they have a number of well known properties that admit efficient inference algorithms. Exponential family distributions have the form:

$$p(x|\phi) = g(x) \exp\{\phi^T F(x) + a(\phi)\}, \tag{2.3}$$

where $F(x)$ is a fixed length vector *sufficient statistic*, ϕ is the *natural parameter* vector, and $a(\phi)$ is a scalar valued function of ϕ that ensures that the distribution normalizes to 1. The exponential family includes the Gaussian, Multinomial, Beta, Gamma, and other common distributions.

We also require an additional restriction that the component prior distribution H

be conjugate to the component distributions [BS94]. It must be of the form:

$$H = p(\phi|\nu, \eta) = h(\eta, \nu) \exp\{\phi^T \nu + \eta a(\phi)\}. \tag{2.4}$$

η and ν are the natural parameters for the conjugate prior distribution.

The following fact is fundamental to our approach: If a set of observations X are all assigned to the same mixture component ($z_i = k$ for all i such that $x_i \in X$), then the posterior distribution of the component parameter ϕ_k is determined by

$$S = \sum_{x_i \in X} F(x_i), \tag{2.5}$$

which is the sum of the sufficient statistic vectors of each observation $x_i \in X$. The significance of this fact is that if a set of assignment variables are constrained to be equal (i.e., their corresponding observations are assumed to be generated by the same mixture component), their inferential impact can be fully summarized by S, a vector whose length does not increase with the number of observations.

2.4 Existing Approaches

We briefly review existing approaches for online learning of Bayesian Mixture Models. Existing approaches have in common that they explicitly consider a number of alternative clusterings or mixture models in parallel, and update each of these hypotheses independently as new data arrives.

2.4.1 Online Variational Bayes

Sato [Sat01] derives recursive update rules for Variational Bayesian learning of mixture models. The alternative models are stored in memory, and each data point is discarded after it is used to update each parallel hypothesis. A "forgetting factor" is used in order to decay the contribution of "old" data points, since they are likely to be incorrectly assigned to components. Empirically, the forgetting factor means that

much more data is needed to learn a model when compared with a batch technique. This makes the forgetting factor undesirable from the standpoint of our requirement to have an incremental algorithm that outputs results substantially close to the results that a batch algorithm would output given the total data seen. Finally, model parameters must be stored for each alternative hypothesis, and this becomes prohibitively expensive as the number of models increases.

2.4.2 Particle Filters

Fearnhead [Fea04] developed a particle filter learning algorithm for the DPMM. This approach approximates $p(Z^T|X^T)$ with a set of M weighted particles (clustering hypotheses). Upon arrival of a new data point, the M particles are extended to include a new assignment z_{T+1} and none of the assignments for the previous observations change. In order to prevent combinatorial explosion over time, only M of these descendant particles are retained. In our experiments, this approach behaves poorly for large datasets. Unseen observations can have a drastic effect on the relative rankings of the assignments Z^T. The algorithm greedily keeps only the top ranked clusterings at time T, and those that it discards can never be considered in the future. No two particles are identical, but the assignments tend to vary from one another for only a small number of data points and so do not cover a wide enough set of hypotheses.

2.5 Our Approach

We observe that the chief difficulty with existing approaches is that they must explicitly enumerate and update a very large number of alternative clusterings in order to produce accurate results (the number of potential clusterings of N points grows as the Bell number B_N). We wish to avoid this explicit enumeration, while at the same time keeping a large number of alternatives alive for consideration. Our approach must also require bounded time and space requirements to produce an update given new data: the computational requirements must not scale with the total number of data seen.

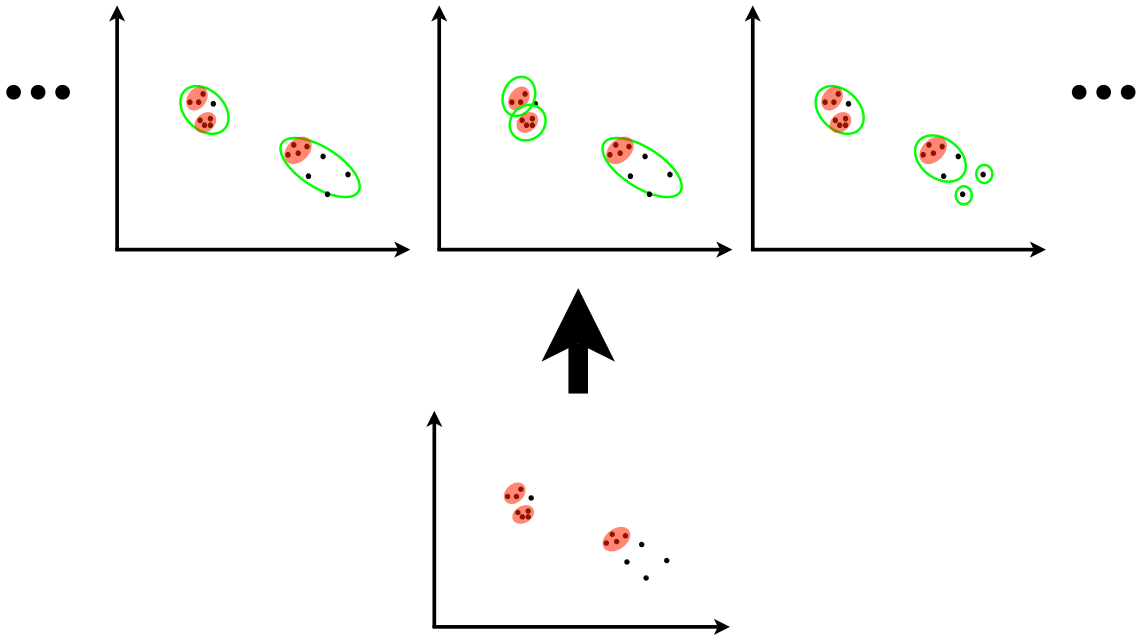


Figure 2.2: A schematic depiction of a two-dimensional clustering problem. Alternative clustering solutions are displayed on their own coordinate axes, and the model clusters are represented by green ellipses. The set of clump constraints consistent with all solutions are displayed as red ellipses. The implicit set of clustering solutions under consideration are those that can be composed of the clumps, which is much larger than other online clustering methods that explicitly enumerate alternative clustering hypotheses.

Figure 2.2 shows how multiple clustering hypotheses can be combined into a single set of assignment constraints. Rather than explicitly fixing the assignments in each parallel branch, the constraints now take the form of points that are grouped together in every alternative. We will call these groups of points "clumps". We define sets of indices C_s such that if $i \in C_s$ and $j \in C_s$ for some s, then data points x_i and x_j are assigned to the same component in all of the alternative clusterings. The sets C_s are disjoint, meaning that no data point can exist in more than one clump. The collection of clumps C is the partition with the fewest number of sets that can compose each of the alternative clustering hypotheses. In the language of lattice theory, the clump partition C is the greatest lower bound or infimum of the alternative clustering hypotheses. A similar scheme was pursued in [BFR98] for scaling the k-means algorithm (where the number of clusters is assumed to be known) to large databases.

A single optimization procedure done under the clump constraints will yield the best clustering mode (modulo local minima issues) that is compatible with the *implicit* ensemble of alternatives inherent in the constraints. The implicit ensemble of alternatives is very large; it is composed of every possible grouping of the clumps, and is much larger than could be explicitly modeled.

This raises the question: How can these clump constraints be computed without first explicitly computing a number of plausible solutions? We observe that alternative models, while distinct, have considerable redundancy. The reason is that the clustering of data points in one region of space has little impact on the clustering assignments of data in a distant part of space. Any two alternatives will tend to vary from one another only for a subset of data points. Our approach is to partition the clustering problem into a series of independent subproblems. This is carried out in a top down fashion, as illustrated in Figure 2.3. This forms a tree of possible groupings of data, and the bottom level of this tree defines our clump constraints. Variational Bayes techniques provide a convenient framework for carrying out this procedure (see Section 2.5.2).

Our algorithm processes data in small batches which we refer to as *epochs*, each

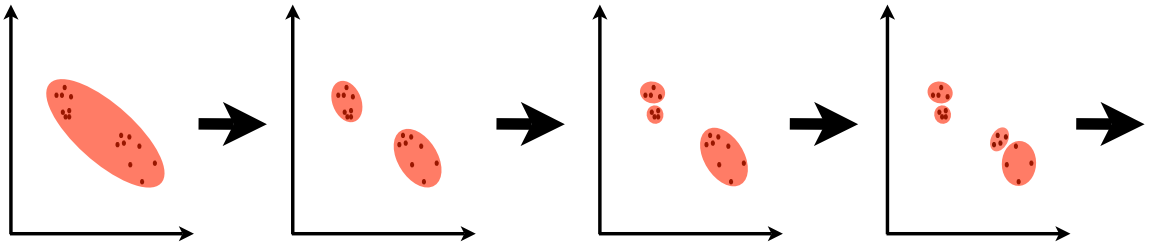


Figure 2.3: Top down compression: Clump constraints are discovered by recursive splitting in a top down fashion. This process is well suited to discovering groups of points that are likely to belong to the same model cluster across multiple plausible clustering alternatives.

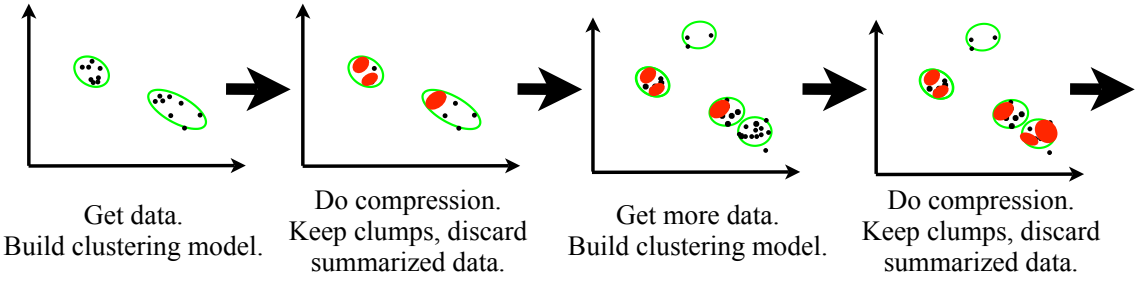


Figure 2.4: A sketch of the algorithm. Data arrives in small batches or epochs, and the current best estimate mixture model is computed in the model building phase (here, the model is represented by green ellipses). Next, clump constraints are computed in the compression phase, and summarized by their associated sufficient statistics (represented by red ellipses). The summarized data are discarded and the process continues.

one of which contains E data points. We first compute the current best estimate mixture model as described in Section 2.5.1. Then we carry out a compression phase (explained in Section 2.5.2) in which clump constraints are computed in a top down recursive fashion, and this phase halts when a stopping criterion is met. Data points that belong to the same clump are summarized by their average sufficient statistics (see Section 2.3.2), and the E individual data points are discarded. The clumps are each given an assignment variable z_s and can be treated in the same way as data points in the next round of learning. We bound the computational time and space requirements in each learning round by controlling the number of clumps discovered during the compression phase. The algorithm is summarized in Figure 2.4.

2.5.1 Model Building Phase

Learning rounds begin by computing a current best estimate mixture model using Variational Bayes (VB) [Att99]. In the Variational Bayes approach, intractable posterior distributions are approximated with simpler proxy distributions that are chosen so that they are tractable to compute. Blei and Jordan [BJ05] extended this technique to the DPMM.

Given the observed data X^T, the batch VB algorithm optimizes the variational Free Energy functional:

$$\mathcal{F}(X^T; q) = \int_{dW} q(V, \Phi, Z^T) \log \frac{p(V, \Phi, Z^T, X^T | \eta, \nu, \alpha)}{q(V, \Phi, Z^T)}, \tag{2.6}$$

which is a lower bound on the log-evidence $\log p(X^T | \eta, \nu, \alpha)$. The proxy distributions

$$q(V, \Phi, Z^T) = \prod_{k=1}^{K} q(V_k; \xi_{k,1}, \xi_{k,2}) q(\phi_k; \zeta_{k,1}, \zeta_{k,2}) \prod_{t=1}^{T} q(z_t) \tag{2.7}$$

are products of beta distributions for the stick-breaking variables (with hyperparameters ξ), component distributions (with hyperparameters ζ), and assignment variables, respectively. Update equations for each proxy distribution can be cycled in an iterative coordinate ascent and are guaranteed to converge to a local maximum of the free energy. The true DPMM posterior allows for an infinite number of clusters, but the proxy posterior limits itself to K components. Kurihara et al. [KWV07] showed that K can be determined by starting with a single component, and repeatedly splitting components as long as the free energy bound $\mathcal{F}(X^T; q)$ improves.

Like the batch approach, our algorithm optimizes $\mathcal{F}(X^T; q)$ during model building, but this optimization is carried out under the clump constraints discovered during previous learning rounds. The resulting Free Energy bound is a lower bound on the optimal batch solution. (In practice, the batch process itself may not achieve the optimal bound because of local optima issues.) Formally this can be expressed as:

Property 1. *The MB-VDP model-building phase optimizes $\mathcal{F}(X^T; q)$ subject to the*

constraints that $q(z_i) = q(z_j)$ *for all* $i \in C_s$ *and all* $j \in C_s$ *and all clump constraints* C_s*. The resulting solution lower bounds the optimal batch solution:* $\max_q \mathcal{F}_{MB}(X^T; q) \leq \max_{q'} \mathcal{F}(X^T; q')$.

The bound follows because solutions to the constrained problem are in the space of feasible solutions of the unconstrained optimization problem. Hyperparameter update equations that optimize the constrained Free Energy $\mathcal{F}_{MB}$ are:

$$\begin{aligned}
\xi_{k,1} &= 1 + \sum_s |C_s| q(z_s = k) \\
\xi_{k,2} &= \alpha + \sum_s |C_s| \sum_{j=k+1}^{K} q(z_s = j) \\
\zeta_{k,1} &= \eta + \sum_s |C_s| q(z_s = k) \langle F(x) \rangle_s \\
\zeta_{k,2} &= \nu + \sum_s |C_s| q(z_s = k) \\
q(z_s = k) &\sim \exp(S_{sk})
\end{aligned} \tag{2.8}$$

where we define

$$S_{sk} = E_{q(V,\phi_k)} \log\{p(z_s = k|V) p(\langle F(x) \rangle_s | \phi_k)\} \tag{2.9}$$

$$\langle F(x) \rangle_s = \frac{1}{|C_s|} \sum_{i \in C_s} F(x_i) \tag{2.10}$$

which (critically) depend only on $\langle F(x) \rangle_s$, the sufficient statistics of the points in each clump.

After executing the update equations for $q(z_s = k)$, the constrained Free Energy may be expressed in the following form:

$$\mathcal{F}_{MB}(\mathbf{X}^T; q) = \mathcal{F}_{\text{likelihood}}(\mathbf{X}^T; q(V), q(\Phi)) - \mathcal{F}_{\text{complexity}}(q(V), q(\Phi)). \tag{2.11}$$

We find that $\mathcal{F}_{MB}$ decomposes into a *likelihood* term that measures the extent to

which the current model fits the compressed data:

$$\mathcal{F}_{\text{likelihood}} = \sum_s n_s \log \sum_{k=1}^{K} \exp(S_{sk}) \tag{2.12}$$

as well as a *complexity* penalty

$$\mathcal{F}_{\text{complexity}} = \sum_{k=1}^{K} KL(q(v_k)||p(v_k|\alpha)) + \sum_{k=1}^{K} KL(q(\phi_k)||p(\phi_k|\lambda)) \tag{2.13}$$

which penalizes models according to the Kullback-Leibler divergence between the model parameters' proxy posterior distributions and their respective prior distributions. The complexity penalty increases with the number of clusters K expressed by the current model. Intuitively, the constrained Free Energy balances the goal of finding models that explain the observed data while preventing overfitting with overly complex models.

The constrained Free Energy $\mathcal{F}_{MB}$ was first given in Kurihara et al. [KWV07], in which DPMM learning is augmented with a kd-tree in order to speed up inference (also [VNV06] for EM learning). Sufficient statistics of data points were cached at nodes of the kd-tree and used to perform approximate inference. Our approach differs from these algorithms in several ways. We do not use a kd-tree to compute clump constraints but instead build a tree by greedily splitting collections of data points according to a Free Energy-based cost function, as discussed in the next section. We process data in sequential rounds and recompute clump constraints after each round. We irreversibly discard individual data points that are summarized by clump statistics in order to maintain storage costs below a pre-assigned bound, whereas [KWV07] and [VNV06] always have the option of working with individual data points if it leads to improvement in a Free Energy bound.

2.5.2 Compression Phase

The goal of the compression phase is to identify groups of data points that are likely to belong to the same mixture component, no matter the exact clustering behavior of

the rest of the data. Once these groups are summarized by their sufficient statistics, they are irreversibly constrained to have the same assignment distribution during future learning rounds. Therefore we must take into account unseen future data when making these decisions in order to avoid locking into suboptimal solutions. We must find collections of points that are not only likely to be assigned to the same component given the first T data points, but also at some target time N, with $N \geq T$.

We estimate this future clustering behavior by using the empirical distribution of data seen so far (up to time T) as a predictive distribution for future data:

$$\hat{p}(\mathbf{x}_{T+1}, \cdots, \mathbf{x}_N) = \prod_{i=T+1}^{N} \frac{1}{T} \sum_{t=1}^{T} \delta(\mathbf{x}_i - \mathbf{x}_t) \tag{2.14}$$

and define the following modified Free Energy

$$\mathcal{F}_C(\mathbf{X}^T; r) = E_{\hat{p}(\mathbf{x}_{T+1}, \cdots, \mathbf{x}_N)} \mathcal{F}_{MB}(\mathbf{X}^N; r) \tag{2.15}$$

by taking the expectation of the constrained Free Energy $\mathcal{F}_{MB}$ over the unobserved future data. We also define a new proxy distrbution $r(V, \Phi, Z^N)$ used during the compression phase, which is identical in form to Eq. 2.7 estimated during the Model Building phase.

Proposition 1. *Iteration of the following parameter update equations results in convergence to a local maximum of $\mathcal{F}_c$:*

$$\begin{aligned}
\xi_{k,1} &= 1 + \frac{N}{T} \sum_s |C_s| r(z_s = k) \\
\xi_{k,2} &= \alpha + \frac{N}{T} \sum_s |C_s| \sum_{j=k+1}^{K} r(z_s = j) \\
\zeta_{k,1} &= \eta + \frac{N}{T} \sum_s |C_s| r(z_s = k) \langle F(x) \rangle_s \\
\zeta_{k,2} &= \nu + \frac{N}{T} \sum_s |C_s| r(z_s = k) \\
S_{sk} &= E_{r(V,\phi_k)} \log\{p(z_s = k|V) p(\langle F(x) \rangle_s | \phi_k)\}
\end{aligned} \tag{2.16}$$

$$r(z_s = k) \sim \exp(S_{sk}).$$

After performing the updates for $r(z_s = k)$, it holds that:

$$\mathcal{F}_C = \left(\frac{N}{T}\right) \mathcal{F}_{likelihood}(\mathbf{X}^T; r(V), r(\Phi)) - \mathcal{F}_{complexity}(r(V), r(\Phi)). \tag{2.17}$$

The above update equations differ from those in the model building phase by a data magnification factor $\frac{N}{T}$. The indices s range over the current clump constraints; we need not compute assignment distributions $r(z_s = k)$ for unobserved future data. We also find that the compression phase objective can be interpreted as re-scaling the data likelihood term by $\frac{N}{T}$.

As indicated in Figure 2.3, we compute clump constraints in a top down fashion. We start the process with the clustering estimate determined during the preceding model building phase; that is, $r(z_s = k) = q(z_s = k)$. We then evaluate splitting each partition k by first splitting it along the principal component defined by the clumps in the partition. We then iterate the update equations (Eqs. 2.16) in order to refine this split. Each potential partition split is then ranked according to the resulting change in $\mathcal{F}_C$ (Eq. 2.15). We then greedily choose the split that results in the largest change. The process repeats itself, until a halting criterion is met (see below). We update the clump constraints according to $C_l = \{s : \text{argmax}_k\, r(z_s = k) = l\}$.

Property 2. *The maximum attainable Free Energy during the MB-VDP model building phase increases monotonically with the number of clump constraints discovered during the compression phase.*

The reasoning is similar to Property 1. Each time the compression phase splits an existing partition into two, the space of feasible solutions in the model building optimization problem has been increased, but the previous set of solutions (all data in the two new partitions constrained to have equal assignment distributions) is still available. Therefore, the maximum attainable Free Energy cannot decrease.

We must restrict the number of clumps that are retained in order to ensure that the time and space complexity is bounded in the next round of learning. A stopping

Algorithm 1 Memory Bounded Variational DPMM

```
while There is more data do
  Model building phase according to sec. 2.5.1
  Initialize compression phase: r(z_s = k) = q(z_s = k)
  while MC < M (eq. 2.18) do
    for k = 1 to K do
      Split partition k and refine (eqs. 2.16)
      S(k) = ΔF_C (change in eq. 2.17)
    end for
    Split partition arg max_k S(k)
    K = K + 1
  end while
  C_l = {s : argmax_k r(z_s = k) = l}
  Retain clump statistics ⟨F(x)⟩_l into next round
  Discard summarized data points
end while
```

criterion determines when to halt the top down splitting process. A number of criteria are possible, depending on the situation.

When learning DPMMs with full-covariance Gaussian components, each clump requires $\frac{d^2+3d}{2}+1$ values to store sufficient statistics (mean, symmetric covariance matrix, and number of data points summarized). It is convenient to express the stopping criterion as a limit on the amount of memory required to store the clumps. From this perspective, it makes sense to replace a clump with its sufficient statistics if it summarizes more than $\frac{d+3}{2}$ data points. If a clump summarizes fewer points, then the individual data points are retained instead. We refer to these individual retained data points as singlets. The clump memory cost for mixture models with full covariance matrices is therefore

$$MC = \left(\frac{d^2+3d}{2}+1\right) N_c + dN_s, \tag{2.18}$$

where N_c is the number of clumps and N_s is the number of singlets. An upper limit on clump memory cost M is defined, and the compression phase halts when $MC \geq M$.

The MB-VDP algorithm is summarized in Algorithm 1. The time required for the algorithm to learn the entire data set is typically less than the batch variational

DPMM approach outlined in [KWV07]. This is because full variational updates in the batch procedure require $O(KN)$, where K is the number of clusters and N is the number of data points. The MB-VDP algorithm requires only $O(K(N_c + N_s + E))$ for an iteration during the model building phase.

We have implemented a number of measures in the compression phase in order to reduce computational overhead. The first is to hard assign clumps to partitions, i.e., $r(z_s) = \delta(z_s - a_s)$ where $a_s = \text{argmax}_k S_{sk}$, rather than maintaining full assignment distributions. The second is to refine split partitions by optimizing $\mathcal{F}_c$ restricted only to the data and parameters associated with the partition, rather than performing complete updates with all data and parameters (this also allows us to cache the candidate partition splits rather than re-computing them during each iteration). When these speed up heuristics are in place, the compression phase can no longer be interpreted as optimization of $\mathcal{F}_C$, however experimental results in Section 2.6 show that the algorithm performs well and that the time required during the compression phase is quite modest when compared to the model building phase.

Vasconcelos and Lippman [VL99] learn a hierarchy of EM mixture model solutions using a bottom up procedure (although they did not investigate this approach in the context of incremental learning). We find that a bottom up approach to learn clump constraints is inappropriate in our situation. Variational updates for the DPMM are sensitive to initial conditions, and our top down method sidesteps this initialization problem.

Our implementation of MB-VDP may be found at: `http://vision.caltech.edu/~gomes`.

2.6 Experimental Results

We test our algorithm with three experiments. The first experiment compares our algorithm against the particle filter in [Fea04] on a small image clustering task of four categories from Caltech 256. The second experiment compares our algorithm against [KWV07] on the larger MNIST digit dataset. Finally, we demonstrate our

approach on $330K$ image patches from the Corel image database, which was too large for the batch approach.

The first set of experiments compares the performance of our method with that of Fearnhead's particle filter. The data set consists of four categories (Airplanes, Motorbikes, Faces, and T-Shirts) from Caltech 256 [GHP07] that are projected to a 20 dimensional feature space using Kernel PCA with the Spatial Pyramid Match Kernel of Lazebnik et al. [LSP06]. There are 1400 data points (images) in total. The hyperparameters for Normal Inverse Wishart prior on cluster parameters (H) were chosen by hand, based on prior knowledge about the scale of the data, and the concentration parameter α was set to 1. The batch algorithm tends to find 12 to 15 clusters in this setting. The clusters discovered respect the categories, that is, very few objects from different classes are clustered together. This was tested by assigning labels to clusters by looking at five examples from each. Images from the training set were classified according to the label of the cluster with highest responsibility. Average classification performance was 98%. However, the algorithm divides each category into sub-categories according to perceptually relevant differences. Figure 2.5 shows some example images from six of the discovered clusters.

The algorithms were judged quantitatively according to predictive likelihood. 1300 of the 1400 images were chosen at random as a training set, and the algorithm is trained on a complete pass through the data in random order. The average likelihood of the remaing data points was computed as a measure of generalization performance. The particle filter was tested at different numbers of particles. The amount of memory was varied for our algorithm. In our algorithm, the memory value represents the memory required to store both the clumps from earlier rounds of memory and the current small batch of points. In all cases, the data epoch size E are chosen to be one-half of the memory size, so for an effective memory of 200, the algorithm progesses through the data in epochs of 100 points. Note that at memory of 200, the algorithm is unable to store all 12 to 15 clusters inherent in the data.

Table 2.1 shows the performance of the particle filter, and Table 2.2 shows the performance of our algorithm. Our algorithm beats the particle filter in terms of

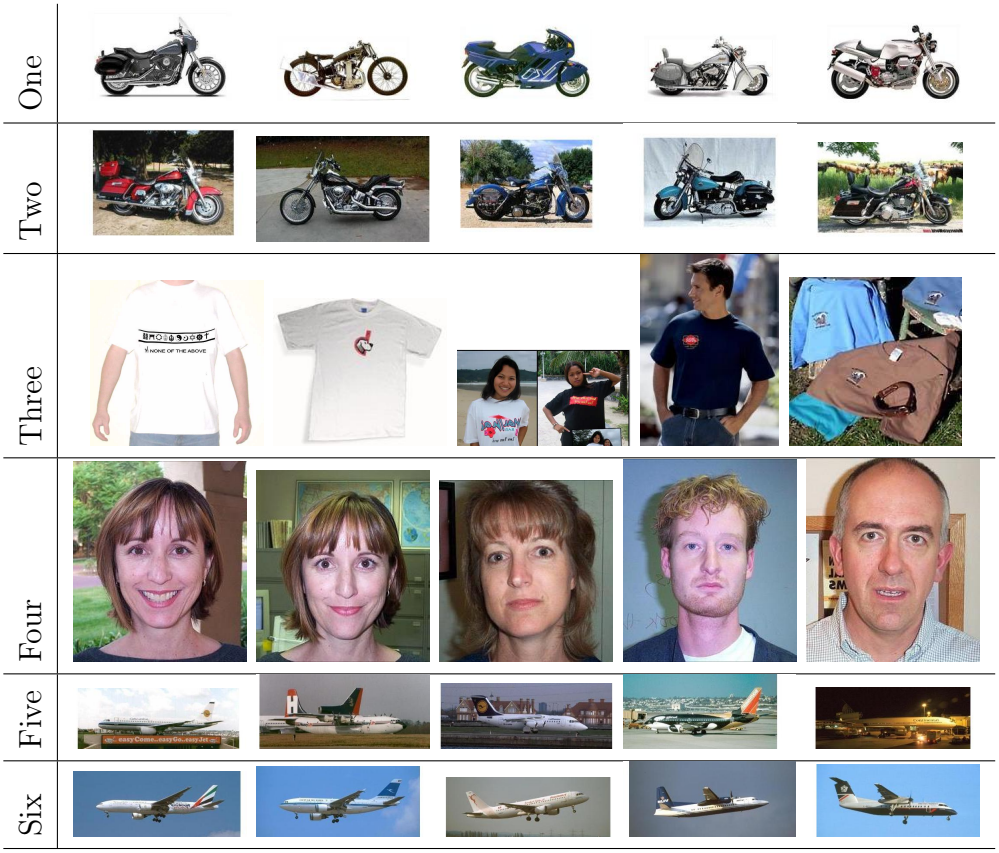


Figure 2.5: Example images from some clusters discovered in the T-Shirts, Airplanes, Faces, and Motorbike categories from Caltech 256. The clusters typically do not mix images from different categories and the algorithm discovers relevant distinctions within categories. For example, the Airplanes category is split into airplanes in the sky and on the ground, and the Motorbikes category is split into segmented motorbikes and motorbikes in clutter.

Particles	Ave Predictive Log-Likelihood	Runtime
100	4.99 ± 0.34	5.94×10^2 s
1000	5.43 ± 0.28	2.856×10^3 s
10000	5.80 ± 0.22	2.484×10^4 s

Table 2.1: Particle filtering predictive performance and runtime

Memory	Ave Predictive Log-Likelihood	Runtime
200	6.37 ± 0.32	73.3 s
400	6.93 ± 0.32	57.08 s
600	6.99 ± 0.31	57.76 s

Table 2.2: MB-VDP predictive performance and runtime. Batch performance was 7.04 ± 0.28 with runtime $71.4s$ on 1300 data points.

generalization accuracy at all parameter values. Our algorithm produces generalization results that are close to the performance of the batch algorithm. The runtime advantage of our approach is very significant over that of the particle filter.

In the second experiment, our approach is compared against the batch algorithm of [KWV07]. The 60000 hand-written digits from the MNIST training set were reduced to 50 dimensions using PCA in a preprocessing step. Our algorithm was set to have a memory size equivalent to 6000 data points, which is an order of magnitude smaller than the size of the data set. Our algorithm processes data in epochs of 3000.

The second row of Figure 2.1 shows the cluster means discovered by our algorithm as it passes through more data. Since the DPMM is nonparametric, the model complexity increases as more data is seen. The bottom row of Figure 2.1 shows the cluster centers discovered by our approach after processing the entire data set compared to those produced by the batch algorithm. The clusters are qualitatively quite similar, and the two algorithms discover a comparable number of clusters (88 for the batch approach, 90 for our algorithm).

The run time for the batch algorithm was 31.5 hours, while for our approach it was 20 hours for a complete pass through. Note that we can likely achieve greater speedup by initializing each learning round with the previous round's model estimate and using a split-merge procedure [UNGH99], although we did not pursue this here.

We compare the free energy bounds produced by the two approaches. The ratio of these two values is 0.9756 meaning that our incremental algorithm produces a slightly worse lower bound on the likelihood. Our approach is more accurate than the kd-tree accelerated algorithm in [KWV07] which produced a free energy ratio of 0.9579 relative to the standard batch approach. Recognition was performed on 10000 MNIST test digits, in the same way as the Caltech 4 dataset but labels were assigned by observing only the cluster means. Performance for the incremental algorithm was 88.5% and 91.2% for batch. Note that this approach only requires labeling of approximately 90 images, compared to 60000 training labels used by traditional approaches.

Finally, we demonstrate our algorithm on a clustering task of 330,000 7 pixel by 7 pixel image patches from the Corel image database. We preprocess the data by discarding patches with standard deviation below a threshold, and normalize all remaining patches to unit norm. We use Gaussian components with diagonal covariance matrices. The batch approach in [KWV07] was unable to cluster this data due to memory requirements. We use an effective memory size of 30000 data points. Cluster centers are shown in Figure 2.6 after 30K, 150K, and 330K patches were processed. As expected, the model complexity increases as more data is processed and the clusters represent greater diversity in the data. The total memory required by the incremental algorithm was 109 MB to store the best estimate model, the clumps, and the responsibilities. In contrast, the batch approach would require 773 MB. The incremental algorithm required approximately 2 hours per epoch of 15000 data points. Again this could be substantially reduced by initializing each round with the previous estimate, rather than beginning from scratch each time.

2.7 Discussion and Conclusions

We have introduced an incremental clustering algorithm with a number of favorable properties. The key idea (summarized by Figure 2.3) is to find clustering arrangements (clumps) that alternative models are likely to have in common, rather than to

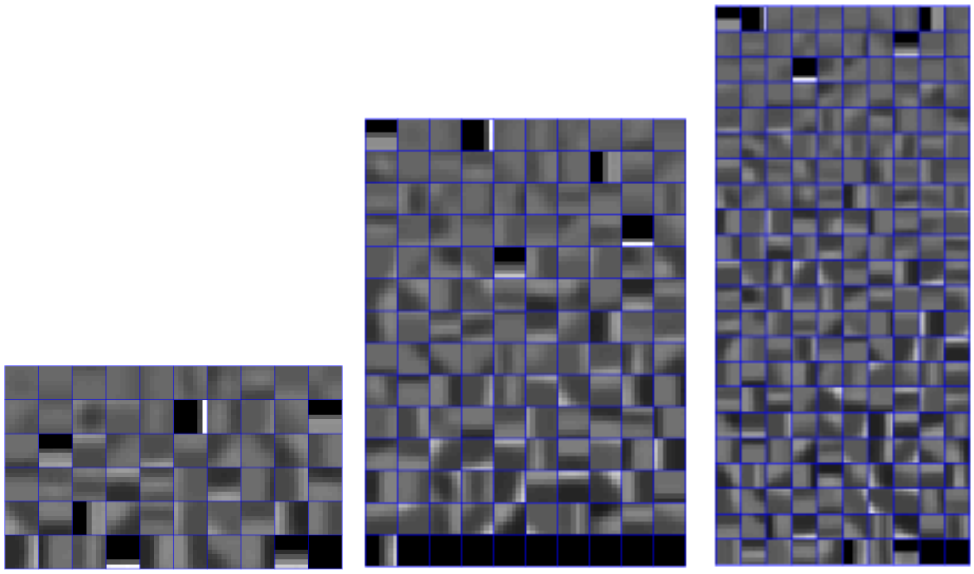


Figure 2.6: Cluster centers from the Corel patch experiment after 30K, 150K, and 330K patches

explicitly enumerate and independently update a set of alternatives. This idea leads to an algorithm that outperforms other online approaches in terms of run time and accuracy, and is suitable for use on large datasets. Our algorithm's nonparametric Bayesian framework allows for automatic determination of the number of clusters, and model complexity adjusts as more data is acquired. Future work includes extending these lessons to build systems capable of learning complex object categories incrementally and with little human supervision.

Chapter 3

Memory-Bounded Inference in Topic Models

3.1 Abstract

What type of algorithms and statistical techniques support learning from very large datasets over long stretches of time? We address this question through a memory-bounded version of a variational EM algorithm that approximates inference in a topic model. The algorithm alternates two phases: "model building" and "model compression" in order to always satisfy a given memory constraint. The model building phase expands its internal representation (the number of topics) as more data arrives through Bayesian model selection. Compression is achieved by merging data-items in clumps and only caching their sufficient statistics. Empirically, the resulting algorithm is able to handle datasets that are orders of magnitude larger than the standard batch version.

3.2 Introduction

Consider a collection of surveillance cameras monitoring at an airport. The cameras learn a model of their environment without supervision. Moreover, they learn for many years without significant interruption. Gradually, as more data is captured, the cameras build a joint model of visual object categories.

This problem is akin to the way children learn to understand the world through

the *continuous* process of mostly unsupervised learning. As children grow up they build an increasingly sophisticated internal representation of object categories that continuously restructures itself.

In this paper we ask ourselves: What statistical techniques are suitable for this Open Ended learning task? First, we need a class of models that can naturally expand as more data arrives, i.e., its capacity should not be bounded a priori. Second, these models should allow efficient learning algorithms, both in terms of time and space. For instance, we should not have to store every single piece of information that has been captured. Our technique must produce a sequence of model estimates that reflect new information as it arrives, and the time required to produce each model update must scale modestly as more data is acquired. Finally, we require that the sequence of learned models are sufficiently similar to those that would be produced by a batch algorithm with access to the entire history of data observed at the time of each model update.

Nonparametric Bayesian techniques such as the Dirichlet Process (DP) [Fer73] and the Hierarchical Dirichlet Process (HDP) [TJBB06] satisfy our first desideratum, in that they naturally increase their model complexity with the available data. However, most existing nonparametric Bayesian approaches are batch algorithms: they require every single data-point to be stored and revisited during learning. A batch algorithm could be naively applied to the continuous learning scenario, but all data would need to be cached and a new batch learning process would be run on the entire dataset to produce each model update. This would violate our second criterion in that the time and space requirements would increase unacceptably as the system ages.

Here we propose a more flexible setup, where we impose a bound on the available memory but still allow the model order to increase with more data. We *compress* the data and the internal representation of the model without losing much in terms of model accuracy. The effect is that time and space requirements scale much more gradually over the lifetime of the system. The memory bound does impose a limit on the total capacity of the model, but this trade-off is flexible and can be adjusted online, i.e., as the model is learned. Experiments with a memory-bounded variational

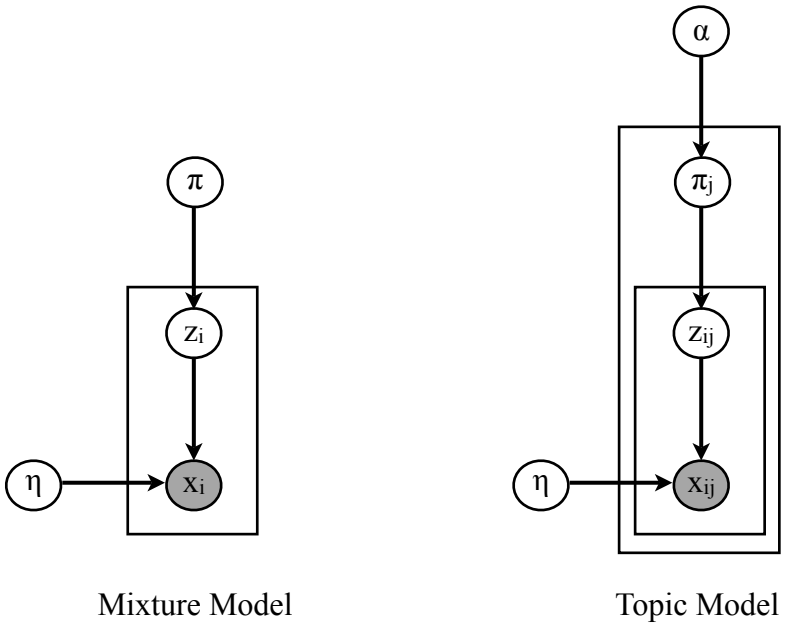


Figure 3.1: Graphical model representations of the mixture model (left) and the topic model (right)

approximation to HDP show that this technique can handle datasets many times larger than the standard implementations and results in substantially shorter runtimes.

3.3 Topic Models

The topic model is an hierarchical extension of the standard statistical mixture model. Figure 3.1 shows the graphical model representations of the mixture model (left) and the topic model (right). The mixture model assumes that observed data $\mathbf{x}_i$ are sampled from one of a (potentially countably infinite) set of component distributions with parameter $\boldsymbol{\eta}_k$. The discrete assignment variable z_i indicates the component that generated $\mathbf{x}_i$. $\boldsymbol{\pi}$ represents the mixture probabilities; that is, the probability that $\mathbf{x}_i$ is sampled from the component distribution with parameter $\boldsymbol{\eta}_k$ (i.e. $z_i = k$) is $\boldsymbol{\pi}_k$.

Whereas the mixture model treats all data points $\mathbf{x}_i$ as being identically distributed from the same mixture (which can be seen by marginalizing out the assign-

ment variables z_i), the topic model assumes that data $\mathbf{x}_{ij}$ are organized in groups indexed by j. For example, $\mathbf{x}_{ij}$ may represent the i-th word in document j, or the i-th pixel or feature in image j. Like the mixture model, each data item is assumed to be a sample from one of a set of component distributions with parameter $\boldsymbol{\eta}_k$. However, the data items in document j are assumed to be drawn from a document-specific mixture model with mixture proportions $\boldsymbol{\pi}_j$. Intuitively, the component distribution parameters $\boldsymbol{\eta}_k$ may be thought of as defining *topics* that are shared across the entire data corpus, and each document j may be modeled as a unique mixture of topics. $\boldsymbol{\alpha}$ represents the proportion of the topics over the corpus as a whole.

The topic models we will explore are most closely related to Latent Dirichlet Allocation (LDA) (introduced by [BNJ03]) and its nonparametric Bayesian extension known as the Hierarchical Dirichlet Process (HDP) [TJBB06] (which allows for a countably infinite set of possible component distributions).

3.4 A Memory-Bounded Variational Topic Model

At a high level the idea is to develop a variational approximation [Att99] for approximating inference in a topic model. We then achieve memory and computational savings by 'clumping' together data-cases. That is, we constrain groups of datapoints to have equal topic assignment variational distributions: $q(z_{ij}) = q(z_{i'j'}) = q(z_c)$ when points x_{ij} and $x_{i'j'}$ are members of the clump c. This allows us to achieve memory savings, because variational optimization performed under this constraint requires only the sufficient statistics of the data-cases in a clump, and the system can forget the exact identities of the summarized data points. Similarly, we will also clump entire documents (or images) by tying their variational distributions over topics: $q(\boldsymbol{\pi}_j) = q(\boldsymbol{\pi}_{j'}) = q(\boldsymbol{\pi}_s)$ if document j and j' belong to the same document group s. This tying of variational distributions guarantees that learning optimizes a lower bound to the exact Free Energy objective function, where the bound is increasingly loose with more tying. This idea was also leveraged in [BFR98] to accelerate the k-means algorithm and in [VNV03] and [KWV07] to accelerate learning mixtures of

Gaussians and DP mixtures of Gaussians by using KD-trees.

In the following we will talk about documents, but we note that this refers to other structured objects such as images as well.

3.4.1 The Variational Topic Model

The following Bayesian topic model is our starting point,

$$p(\mathbf{x}, \mathbf{z}, \boldsymbol{\eta}, \boldsymbol{\pi}, \boldsymbol{\alpha}) = \prod_{ij} p(\mathbf{x}_{ij}|z_{ij}; \boldsymbol{\eta})\ \pi_{j,z_{ij}} \tag{3.1}$$
$$\left[\prod_k p(\boldsymbol{\eta}_k|\boldsymbol{\beta})\right]\left[\prod_j \mathcal{D}(\boldsymbol{\pi}_j; \boldsymbol{\alpha})\right]\left[\prod_k p(\alpha_k)\right]$$

where $\mathbf{x}_{ij}$ is word i in document j and z_{ij} denotes the topic that generated $\mathbf{x}_{ij}$. $\boldsymbol{\pi}_j$ denotes the mixture of topics that generated the words in document j, with $\sum_k \pi_{jk} = 1$. $\boldsymbol{\pi}_j$ are distributed according to a Dirichlet distribution with parameter $\boldsymbol{\alpha}$. Boldface symbols denote vector valued quantities. In this expression we will assume that $p(\mathbf{x}|z, \boldsymbol{\eta})$ is in the exponential family[1],

$$p(\mathbf{x}|z = k, \boldsymbol{\eta}) = \exp\left[\sum_l \eta_{kl}\ \phi_l(\mathbf{x}) - A_k(\boldsymbol{\eta}_k)\right] \tag{3.2}$$

and $p(\boldsymbol{\eta}|\beta)$ is conjugate to $p(\mathbf{x}|z, \boldsymbol{\eta})$,

$$p(\boldsymbol{\eta}_k|\boldsymbol{\beta}) = \exp\left[\sum_l \beta_l \eta_{kl} - \beta_0 A_k(\boldsymbol{\eta}_k) - B(\boldsymbol{\beta})\right]. \tag{3.3}$$

The posterior distributions over $\boldsymbol{\pi}, \boldsymbol{\eta}, \mathbf{z}$ are approximated variationally as

$$q(\boldsymbol{\eta}) = \prod_k q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k) \tag{3.4}$$

[1]Strictly speaking, the exponential family includes additional multiplicative terms $h(\mathbf{x})$ in the expression for $p(\mathbf{x}|\boldsymbol{\eta})$ and $g(\boldsymbol{\eta})$ in the expression for $p(\boldsymbol{\eta}|\boldsymbol{\beta})$. We have left these terms out to simplify the derivation and because for most well known distributions they are simply 1. However, it is straightforward to include them.

$$q(\boldsymbol{\pi}) = \prod_j \mathcal{D}(\boldsymbol{\pi}_j; \boldsymbol{\zeta}_j) \tag{3.5}$$

$$q(\mathbf{z}) = \prod_{ij} q(z_{ij}) \tag{3.6}$$

where we have introduced variational parameters $\{\boldsymbol{\xi}_{kl}, \zeta_{kj}, q_{ijk}\}$, the latter subject to $\sum_k q_{ijk} = 1$. Furthermore, $\mathcal{D}$ denotes a Dirichlet distribution while $q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k)$ is also conjugate to $p(\mathbf{x}|z = k, \boldsymbol{\eta})$,

$$q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k) = \exp\left[\sum_l \xi_{kl}\eta_{kl} - \xi_{k0}A_k(\boldsymbol{\eta}_k) - B_k(\boldsymbol{\xi}_k)\right]. \tag{3.7}$$

By writing down the variational Free Energy and minimizing it over $\boldsymbol{\xi}, \boldsymbol{\zeta}$ we find the following intuitive updates,

$$\xi_{kl} = F_{kl} + \beta_l; \qquad F_{kl} \triangleq \sum_{ij} q_{ijk}\ \phi_l(\mathbf{x}_{ij}) \tag{3.8}$$

$$\xi_{k0} = N_k + \beta_0; \qquad N_k \triangleq \sum_{ij} q_{ijk} \tag{3.9}$$

$$\zeta_{kj} = N_{kj} + \alpha_k; \qquad N_{kj} \triangleq \sum_{i} q_{ijk} \tag{3.10}$$

and

$$q_{ijk} \leftarrow \frac{1}{Z_{ij}} \frac{\exp\left[\sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\ \phi_l(\mathbf{x}_{ij})\right]}{\exp\left[\mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}]\right]}\ \exp\left[\psi(\zeta_{kj})\right] \tag{3.11}$$

where Z_{ij} enforces the constraint $\sum_k q_{ijk} = 1$ and the expectations are over $q(\boldsymbol{\eta})$. To learn the parameters $\{\alpha_k\}$ we first introduce gamma priors,

$$p(\boldsymbol{\alpha}) = \prod_k \mathcal{G}(\alpha_k; a, b). \tag{3.12}$$

Using the bounds in [Min00] we can derive the following updates if we first insert the updates for $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ into the Free Energy,

$$\alpha_k \leftarrow \frac{(a-1) + \alpha_k \sum_j \left[\psi(\zeta_{kj}) - \psi(\alpha_k)\right]}{b + \sum_j \left[\psi(\zeta_j) - \psi(\alpha)\right]} \tag{3.13}$$

with $\zeta_j = \sum_k \zeta_{kj}$ and $N_j = \sum_k N_{kj}$.

3.4.2 Optimizing the Number of Topics K

Our strategy to search for a good value of K is to *truncate* the topic distributions as $q(z_{ij} > K) = 0$ (see also [TKW08]). This will have the effect that most terms in the Free Energy with $k > K$ will cancel, the exception being the prior terms $p(\alpha_k),\ k > K$. For these terms we know that the value for α_k minimizing the Eree Energy is given by the MAP value of the gamma-prior $\alpha_k = \frac{a-1}{b},\ k > K$. Inserting this back into the Free Energy we accumulate $K_{\max} - K$ terms

$$\Lambda = a \log b - \log \Gamma(a) + (a-1) \log \frac{a-1}{b} - (a-1) \tag{3.14}$$

where $K_{\max}$ is the maximum number of topics.

It is guaranteed that there exists a solution with lower Free Energy if we increase K. The reason is that we relax a self-imposed constraint on variational parameters (that $q(z_{ij} > K) = 0$). As K increases the relative improvement in Free Energy quickly attenuates. The final value for K is obtained by thresholding this relative improvement.

The *nesting* property (models with larger K are better) is the same for variational approximations to the DP in [KWV07] and HDP [TKW08]. This raises the question whether we can take the infinite limit for our model as well. The problem is that $(K_{\max} - K)\Lambda \to \infty$ as $K_{\max} \to \infty$. This can be traced back to the fact that we should have added a proper prior $p(K)$ which would have diminished the contribution at large K. Instead we choose an improper, constant prior to avoid the need to estimate likely values for K a priori. However, it is still possible to work with infinite free energies because we are only interested in the relative *change* in Free Energy after increasing K, which is a finite quantity.

In our experiments we chose $a = 1$ and $b = 0.5$, so that the MAP prior value of α_k is 0.

3.4.3 Clumping Data-Items and Documents

We will now tie some of the variational distributions $\{q_{ijk}\}$ across different data-items within and across documents (images) to a 'clump distribution' q_{ck}. Similarly, we will tie some document-specific distributions over topics $\{q(\boldsymbol{\pi}_j)\}$ into a document group $q(\boldsymbol{\pi}_s)$. Note that since we impose constraints on the variational distributions this has the effect of loosening the variational bound.

Define D_s to be the number of documents in a document group, N_c the number of data-items in a word clump, N_{cs} the number of words in document group s and word clump c, and finally $\Phi_l^c \triangleq \sum_{ij \in c} \phi_l(\mathbf{x}_{ij})$. In terms of these we further define,

$$N_{ks} \triangleq \sum_c q_{ck} N_{cs} \tag{3.15}$$

$$N_k \triangleq \sum_c q_{ck} N_c \tag{3.16}$$

$$F_{kl} \triangleq \sum_c q_{ck}\ \Phi_l^c \tag{3.17}$$

With these definitions we derive the following 'clumped' update rules for the variational parameters ξ_{kl} and ζ_{ks},

$$\xi_{kl} = F_{kl} + \beta_l \tag{3.18}$$

$$\xi_{k0} = N_k + \beta_0 \tag{3.19}$$

$$\zeta_{ks} = \tfrac{N_{ks}}{D_s} + \alpha_k \tag{3.20}$$

and

$$q_{ck} \leftarrow \frac{1}{Z_c} \frac{\exp\left[\sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\ \frac{\Phi_l^c}{N_c}\right]}{\exp\left[\mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}]\right]}\ \exp\left[\sum_s \frac{N_{sc}}{N_c}\psi(\zeta_{ks})\right]. \tag{3.21}$$

The update for $\boldsymbol{\alpha}$ becomes

$$\alpha_k \leftarrow \frac{(a-1) + \alpha_k \sum_s D_s \left[\psi(\zeta_{ks}) - \psi(\alpha_k)\right]}{b + \sum_s D_s \left[\psi(\zeta_s) - \psi(\alpha)\right]}. \tag{3.22}$$

An expression for the Free Energy, after inserting expressions 3.18, 3.19, and 3.20, is given by Eq. 3.29 in the appendix.

3.5 Incremental Learning with a Memory Constraint

Our algorithm processes data in small groups composed of E documents, which we refer to as *epochs*. After the arrival of each epoch the algorithm proceeds in two stages: a model building phase during which a new model estimate is produced, and a compression phase in which decisions are made as to which words and documents to clump. The sufficient statistics of each clump are computed and data summarized by clumps are purged from memory. The assignment distributions $q(z)$ of purged data and topic distributions of merged documents $q(\boldsymbol{\pi})$ are discarded as well. The clump sufficient statistics are retained along with the current model estimate, which serves as a starting point for the next round of learning.

3.5.1 Model Building Phase

The model building phase optimizes the Free Energy under the parameter tying constraints induced by the choice of clumps in previous compression phases. We perform a split-merge procedure similar to [UNGH99] to determine the number of topics, using the heuristics in that work to rank topic suitability for split or merge. In our experiments we use Gaussian topic distributions, so splits are proposed along the principal component of the topic. The split proposals are refined by restricted variational updates. That is: Eqs. 3.21, 3.18, 3.19, 3.20, and 3.22 are iterated but only for data-points whose highest responsibility is to the split topic, and the points may be assigned only to the two descendent topics. Merges are carried out by instantiating a new topic with the data-points with highest responsibility to the merged topics. A total of 10 splits and 10 merges are proposed, and evaluated by the resultant change in Free Energy (Eq. 3.29). The top ranked change is then used to initialize full variational updates (which involve all data points). The model building phase halts once the change in Free Energy divided by its previous value is below a threshold,

Algorithm 2 Model Building Phase (Algorithm 3.1)

Input: Previous model $\{\xi_{kl}, \zeta_{ks}, \alpha_k, \Phi^c_{kl}, N_{cs}, D_s\}$, and current epoch of E documents.
Initialize $\zeta_{jk} = \alpha_k$ for $j = |S| + 1, \cdots, |S| + E$
Iterate eqs. 3.21, 3.18, 3.19, 3.20, and 3.22 until convergence
repeat
  Rank splits and merges according to criteria in [UNGH99]
  for $i = 1$ **to** 10 **do**
    Split i-th ranked candidate topic along principal component
    Restricted iteration of eqs. 3.21, 3.18, 3.19, and 3.20 until convergence
    Evaluate change in eq. 3.29 resulting from split
  end for
  for $i = 1$ **to** 10 **do**
    Merge i-th ranked pair of topics
    Evaluate change in eq. 3.29 resulting from merge
  end for
  Select split or merge that yielded largest change in eq. 3.29
  Iterate eqs. 3.21, 3.18, 3.19, and 3.20 until convergence
until Change in eq. 3.29 is less than threshold

which was chosen to be $1E-5$ in our experiments. The procedure is summarized in Algorithm 3.5.1.

3.5.2 Compression Phase

The goal of the compression phase is to determine groups of data-points that are to be summarized by clumps, and to identify documents that are to be merged into document groups.

Clumps are identified using a greedy top down splitting procedure. Because data-points summarized by clumps are ultimately discarded, the compression process is irreversible. Therefore it is of fundamental importance to predict the locations of future data when deciding which points to clump. In order to estimate this, we rank cluster splits according to a modified Free Energy (eq. 3.30) in which the data sample size is artificially increased by a factor $\frac{T_{pts}}{\sum_c N_c}$ and the number of documents is scaled by $\frac{T_{docs}}{\sum_s D_s}$, where T_{pts} and T_{docs} are the target number of data-points and documents expected during the lifetime of the system. This is equivalent to using

the data empirical distribution as a predictive model of future data (see Section 2.5.2 for more information.) If we determine clumps using the standard Free Energy, then the algorithm fails to split large groups of points that are likely to split once more data has arrived. Instead, it wastes memory by placing "stray" points in their own clumps.

We initialize the process by hard assigning each clump or data-point to the cluster with highest responsibility during the previous model building phase. We then proceed through each cluster and split it along the principal component, and refine this split by iterating restricted variational updates equations for the points in the cluster. The updates are modified by the data magnification factors:

$$\xi_{kl} = \left(\frac{T_{pts}}{\sum_c N_c}\right) F_{kl} + \beta_l \tag{3.23}$$

$$\xi_{k0} = \left(\frac{T_{pts}}{\sum_c N_c}\right) N_k + \beta_0 \tag{3.24}$$

$$\alpha_k \leftarrow \frac{(a-1) + \left(\frac{T_{docs}}{\sum_s D_s}\right) \alpha_k \sum_j \left[\psi(\zeta_{ks}) - \psi(\alpha_k)\right]}{b + \left(\frac{T_{docs}}{\sum_s D_s}\right) \sum_s \left[\psi(\zeta_s) - \psi(\alpha)\right]}. \tag{3.25}$$

Updates for q_{ck} and ζ_{ks} are unchanged. After the clusters are refined, the data-points are then hard assigned to the sub-cluster with greatest responsibility, and the proposed split is ranked according to the resultant change in Eq. 3.30. We then greedily split the cluster with highest rank. The process repeats itself, with new clusters ranked in the same way described above. We cache the results of each split evaluation to avoid redundant computation. After we have reached a given memory-bound we extract the partitions resulting from this recursive splitting procedure as our new clumps.

Each clump must store sufficient statistics for full covariance Gaussian components which require $\frac{d^2+3d}{2}$ values, where d is the dimension of the feature space. In addition, $|S|$ (the number of document groups) values must be stored to represent the counts N_{cs} for each clump. Note that from this perspective, it only makes sense to create clumps within a cluster if it contains more than $\frac{d+3}{2} + \frac{1}{d}$ data-points. If not, then it is

Algorithm 3 Clump Compression (Algorithm 3.2)

Input: Output from model building phase: $\{q_{ck}, \Phi^c_{kl}, N_{cs}, D_s\}$, current epoch of E documents and memory-bound M.
Hard partition clumps: $r_c = \arg\max_k q_{ck}$
while $MC < M$ (eq. 3.26) **do**
  for $i = 1$ **to** K **do**
    Split i-th cluster along principal component
    Iterate data magnified restricted updates until convergence
    Hard partition clumps into child clusters
    Evaluate change in eq. 3.30 resulting from split
  end for
  Select split that yielded largest change in eq. 3.30
  $K = K + 1$
end while

more efficient to store the individual data-points and we refer to them as "singlets". The total memory cost of summarizing the data is then

$$MC = \left(\frac{d^2 + 3d}{2}\right)|N_c > 1| + |S||N_c > 1| + d|N_c = 1|, \tag{3.26}$$

where $|N_c > 1|$ is the number of clumps with more than 1 data-item in them, and $|N_c = 1|$ is the number of singlets. The clump compression procedure is summarized in Algorithm 3.5.2.

Document merging provides another way of controlling the memory cost, by reducing the number of image groups $|S|$. We use the following simple heuristic to rank the suitability of merging document groups s and s':

$$DM_{s,s'} = \frac{\sum_k \mathbb{E}[\pi_{sk}]\mathbb{E}[\pi_{s'k}]}{\|\mathbb{E}[\boldsymbol{\pi}_s]\|\|\mathbb{E}[\boldsymbol{\pi}_{s'}]\|}. \tag{3.27}$$

Clumping and document merging enable a number of potential schemes for controlling space and time costs, depending on the application. We note that the time complexity per variational iteration scales as $O(K(|N_c > 1| + |N_c = 1|) + |S|K)$ and the space required to store $q(z_c)$ distributions is $O(K(|N_c > 1| + |N_c = 1|))$.

3.6 Experiments

We test our approach with two machine vision experiments. The first is an image segmentation task, and the second is an object recognition and retrieval task.

3.6.1 Joint Image Segmentation

Our first experient is a joint image segmentation problem. The dataset is the Faces-Easy category of the Caltech 101 image dataset [FFFP04] consisting of 435 images. Each image contains a face centered in the image, but the lighting conditions and background vary. In terms of the vocabulary of the preceding sections, each image is a document and each pixel in the image is a word. Pixels are represented as five dimensional vectors of the following features: X and Y position relative to the center of the image, and three color coordinates in the CIELAB colorspace. The goal of our experiment is to find similar image regions across the multiple images, in an unsupervised way. We emphasize that our main objective is to study the efficiency of our algorithm, not to produce a state of the art image segmentation algorithm.

The images were scaled to be 200 by 160 pixels in size. Thus, the total size of the dataset is 32,000 pixels per image, times 435 images, times 5 features per pixel equals 69,600,000 real numbers. Each pixel requires an assignment distribution. Our baseline implementation (i.e., a batch algorithm that processes all images in memory at once and does not use pixel clumping or image merging) was only able to jointly segment 30 images simultaneously before running out of memory. The majority of memory is used to store the assignment distributions of pixels, and this is problematic as the number of topics increases during learning, since the space requirements scale as $O(NK)$, where N is the total number of pixels and K is the number of topics.

We first compare the memory-bounded approach to the baseline implementation on a joint segmentation task of 30 images in order to judge the impact of the pixel clumping approximation. We vary the upper limit on the number of clumps used to summarize the data during the compression phase, and compare the Free Energy bounds produced by the memory-bounded algorithm to those produced by the

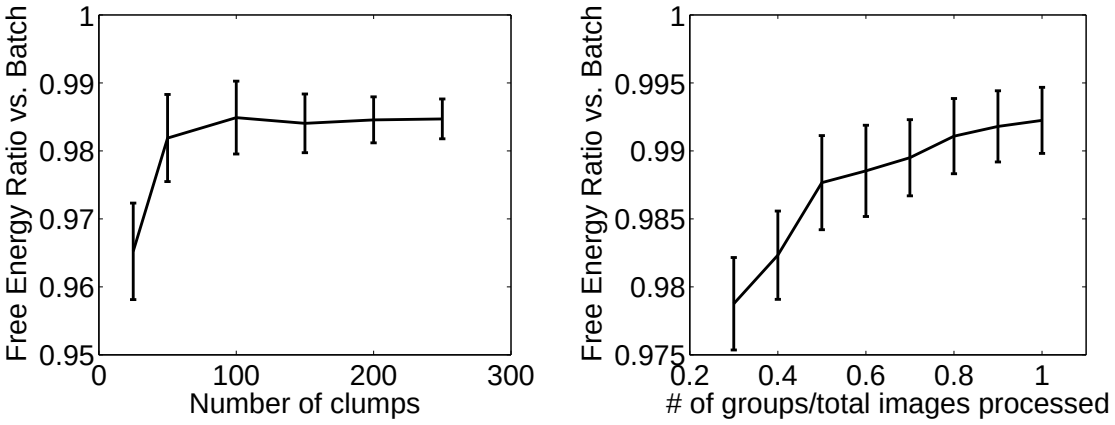


Figure 3.2: Image segmentation experiment. Left: Free Energy ratio as a function of the number of clumps permitted by the memory bound. Right: Free Energy ratio versus the number of image groups relative to the total number of images processed.

baseline implementation. We define the Free Energy ratio as $1 - \frac{FE_{batch} - FE_{mb}}{|FE_{batch}|}$. This process was repeated for different subsets of 30 images from the dataset. In the memory-bounded approach, images were processed in epochs of five images at a time. Figure 3.2 summarizes the results. We find that performance tends to saturate beyond a certain number of clumps.

We also note a significant run time advantage of the memory bounded algorithm over the batch method. The average run time of the batch method was 3.09 hours versus 0.68 hours for the memory-bounded approach.

Next we study the impact of image (document) merges on the relative performance of the memory-bounded algorithm versus the baseline batch algorithm, while varying the maximum number of image (document) groups permitted. The results are shown in Figure 3.2.

We find little qualitative difference between segmentations produced by the baseline and memory-bounded algorithms. The possible exception is in the case when the memory-bounded algorithm is run with a large number of image merges, in which case the algorithm seemed to discover fewer topics than the batch and memory-bounded algorithm with only word clumping. Example image segmentations and clump distributions are shown in Figure 3.3.

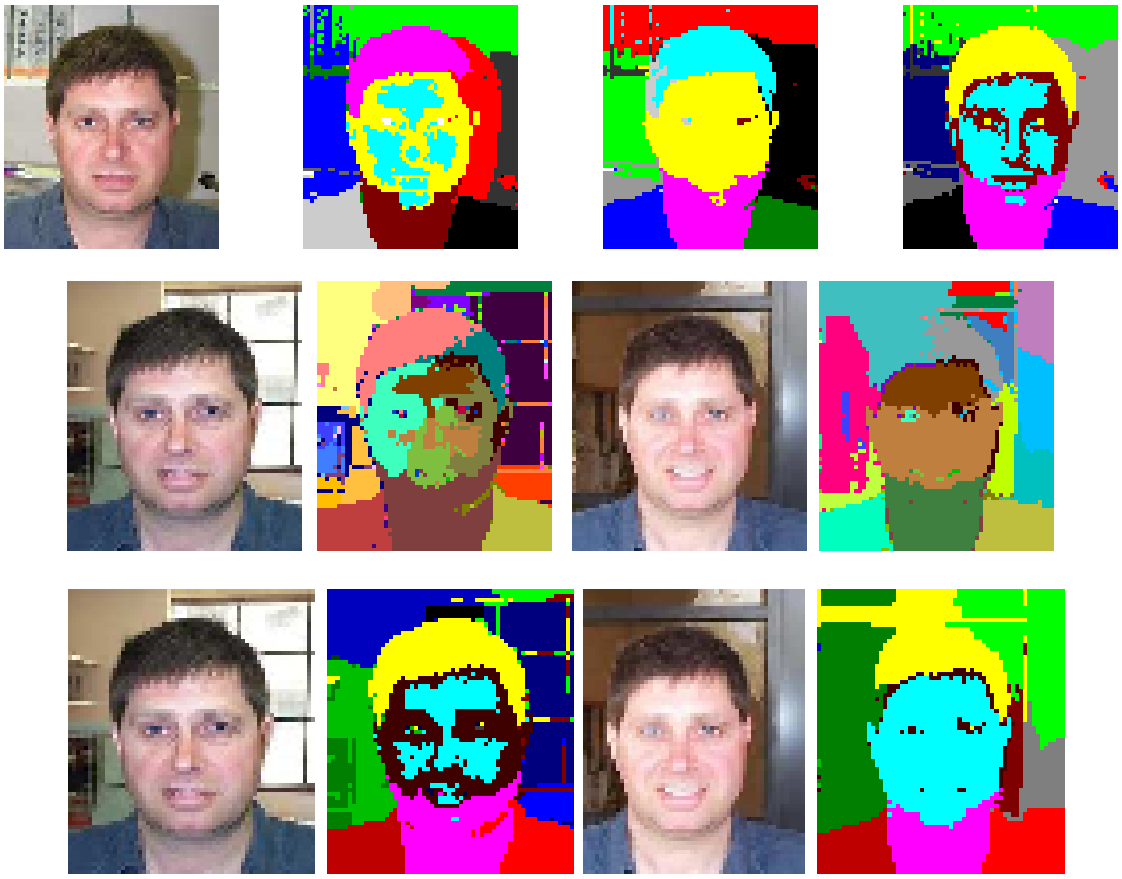


Figure 3.3: Top row: From left to right: an example segmentation produced by the baseline method, memory-bounded algorithm with 30% of total images and 125 clumps, and the memory-bounded algorithm with no images merged and 125 clumps. Row 2: Example clump distributions. Pixels of the same color are summarized in a single clump. Row 3: segmentations corresponding to clumps in row 2.

Finally, we demonstrate the memory-bounded algorithm on the full dataset of 435 images, which is more than an order of magnitude larger than can be handled with the baseline algorithm. We process images in *epochs* of 10 images at a time, for a total of 44 learning rounds. The upper limit on the number of clumps was set to 1000, which was likely many more than required since there were only 85 inferred topics. Because the number of documents was relatively small, we chose not to use document merges. The total run time of the algorithm was 15 hours. Figure 3.4 shows the number of topics as a function of the number of images processed, and the run time required during each image round. The run time is longer during learning rounds in which

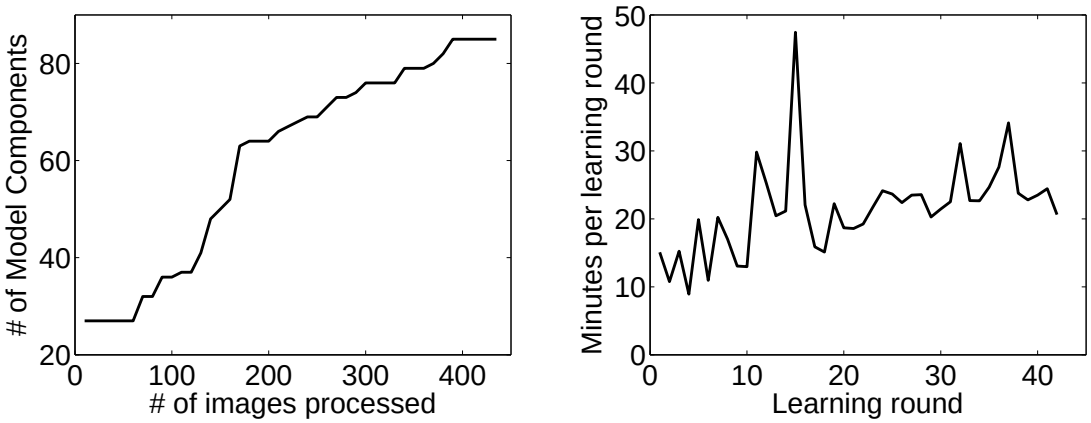


Figure 3.4: Joint segentation of 435 faces. The left plot shows the number of topics recovered as the system processes images. The right plot shows the run time for each learning round. This fluctuates with the number of new topics discovered during each round and tends to increase gradually with the total number of topics.

more new topics are discovered, because more split-merge operations are necessary. The memory required for the memory-bounded algorithm was 22 MB to store the current image epoch and clumps, less than 1MB for the current model estimate, and 235 MB for assignment distributions, for a total of 257 MB. In contrast, the baseline batch implementation would have required 531 MB to store all 435 images, 8.8155 GB to store assignment distributions for each pixel assuming 85 topics, and less than 1 MB for the model, for a total of 9.3 GB. (All memory amounts assume double precision floating point.) The memory-bounded implementation, therefore, achieved a memory savings factor of about 38 with very little loss in accuracy.

Figure 3.5 shows example joint segmentations produced by the memory-bounded algorithm. These images were retrieved by first computing responsibilities for every image in the dataset, with respect to the final model estimate produced by the MB algorithm. Then, the images were sorted according to those that have the most pixels assigned to the largest topic. The largest topic indeed corresponds to a face, and is represented by the olive green segment in the figure. Other topics shared across images include hair and certain backgrounds.

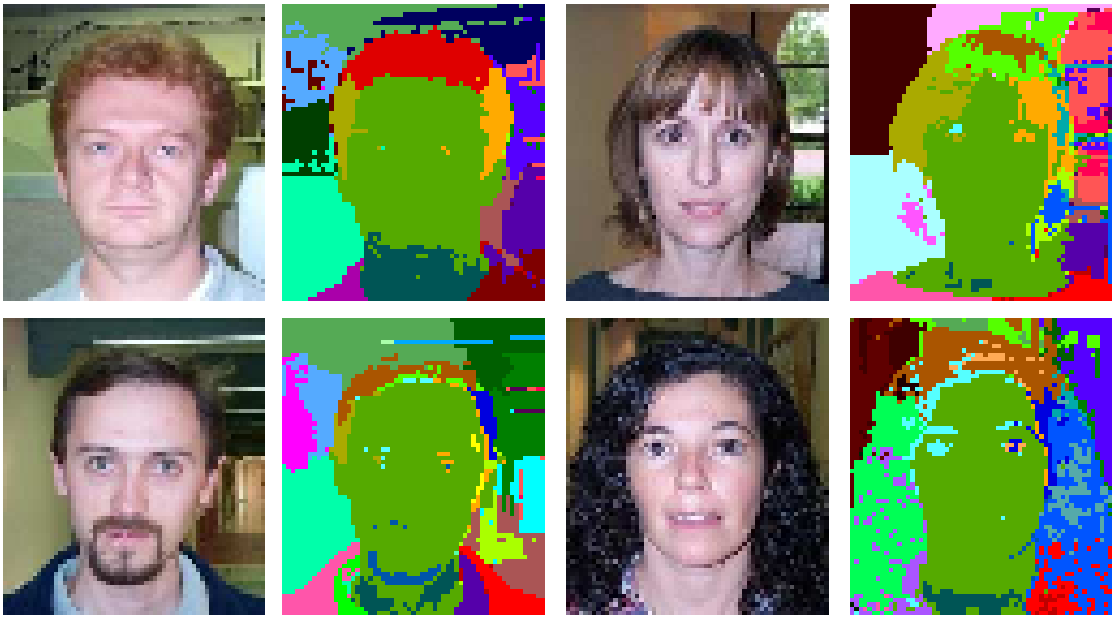


Figure 3.5: Examples of joint segmentation produced after processing all Caltech Face images. Pixels that are the same color have highest responsibility to the same topic. These images were retrieved by sorting images according to those that have the most pixels assigned to the largest topic, which is the olive green colored face segment in each image.

3.6.2 Object Recognition and Retrieval

Our object recognition and retrieval experiment involves all 101 object categories in the Caltech 101 dataset. We randomly select 3000 training images and 1000 test images. We extract 128-dimensional SIFT [Low04] local appearance descriptors from 500 randomly chosen locations in each image. The scale of each feature is also chosen randomly. In the language of topic models, each feature descriptor is a word, and the collection of feature descriptors in an image forms a document. This image representation is known as 'bag-of-features', because images are modeled as unordered collections of feature descriptors whose geometric positions are ignored. This dataset proved too large to compare directly to the batch algorithm

We train a single topic model on all training images, using epochs of 60 images at a time. Because hundreds of topics are discovered we use diagonal covariance Gaussians

and adjust Eq. 3.26 accordingly. Given a test image $\tilde{\mathbf{x}}$, retrieval is performed by ranking each training image's similarity to the test image. To develop the similarity measure we begin with $\log \prod_i p(\tilde{\mathbf{x}}_{ij}|\mathbf{x})$, which is the log-probability that the detections in the test image were generated by training image j given the training set. Then we variationally lower bound this quantity to obtain a test Free Energy and drop all constant terms not involving the test image and index j. Finally we lower bound this quantity by assuming that detections in the test image are hard assigned to the topic with highest responsibility (this leads to an expression that is much faster to evaluate with neglible impact on retrieval performance.) The retrieval score is:

$$\begin{aligned} score(j) = \sum_i \max_k \{ \sum_l & \mathbb{E}[\eta_{kl}|\xi_{kl}] \ \phi_l(\tilde{\mathbf{x}}_{ij}) \\ & - \mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}] + \psi(\zeta_{kj}) \\ & - \psi(\sum_k \zeta_{kj}) \} \end{aligned} \tag{3.28}$$

where the expectations are with respect to $q(\boldsymbol{\eta})$ learned during training and ξ_{kl} and ζ_{kj} are from training as well. ζ_{kj} are re-estimated for images that were merged into a document group during training. We compute nearest neighbor (1-NN) classification accuracy by classifying the test image to the class label of the highest scoring image in the training set.

Figure 3.6 shows the training set Free Energy and 1-NN classfication accuracy as a function of the memory bound M (measured as the equivalent number of data points that could be stored in the same space.) Because we used diagonal covariance matrices, there were enough clumps even at low levels of memory to maintain comparable classification performance. We note that the training Free Energy increases with memory as expected, and that the 1-NN accuracy tends to saturate as memory increases.

Figure 3.7 shows the 1-NN accuracy and training Free Energy when the percentage of document groups relative to the number of total images processed is varied (the memory bound M is held fixed at 10000). We note that the classification performance

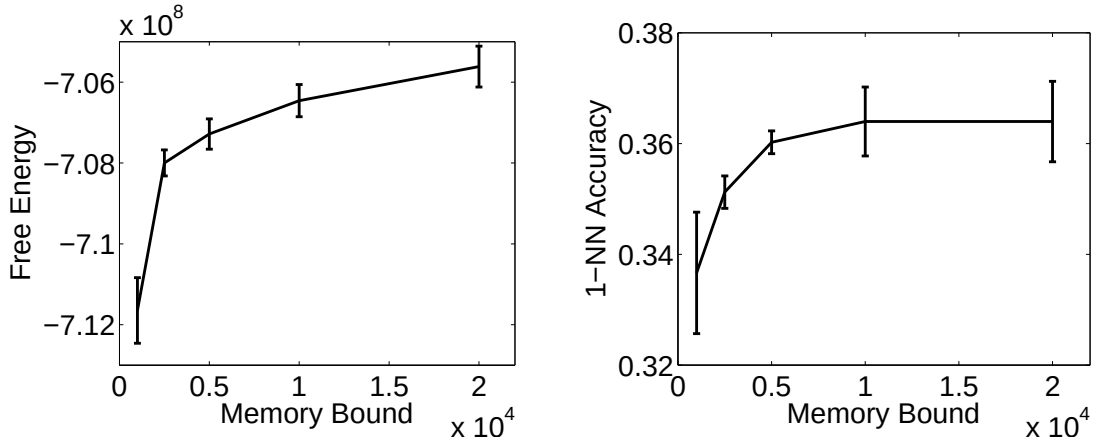


Figure 3.6: Object Recognition and Retrieval. Left: Training set Free Energy as a function of the memory bound. Right: 1-NN classification accuracy as a function of memory bound (measured as the equivalent number of data-points that could be stored in the same space)

suffers substantially when only small numbers of document groups are permitted. We use a heuristic for determining documents to merge (Eq. 3.27). It is possible that a well-motivated criterion (perhaps derived from the Free Energy) would give better performance.

3.7 Conclusion

Machine learning has largely focussed on algorithms that run for a relatively short period of time, fitting models of finite capacity on a data-set of fixed size. We believe that this scenario is unrealistic if we aim at building truly intelligent systems. We have identified nonparametric Bayesian models as promising candidates that expand their model complexity in response to new incoming data. The flip-side is that non-parametric Bayesian algorithms are 'example-based' and as such require one to cache and process repeatedly every data-case ever seen. The objectives of infinite, adaptive model capacity on the one hand, and efficiency, both in time and space, on the other therefore seem to be fundamentally at odds with each other.

In this paper we have made a first step towards resolving this issue by introducing a class of models that can adapt their model complexity adaptively but are able to do

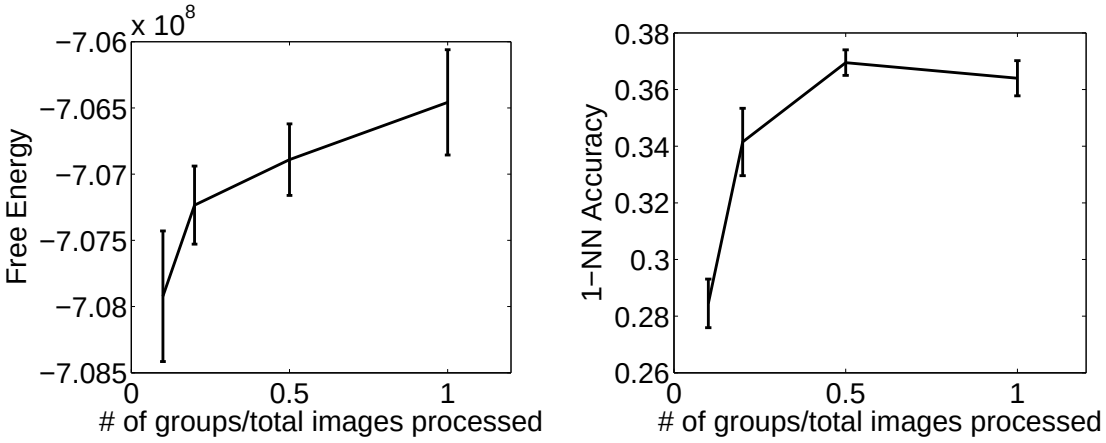


Figure 3.7: Object Recogniton and Retrieval. Left: Training set Free Energy versus the ratio of document groups to the total number of images processed. Right: 1-NN classification accuracy versus the ratio of document groups to total number of images processed

so at a fraction of the memory requirements and processing times necessary for their batch counterparts. There is no magic of course: with a fixed memory budget there is a limit to how complex the model can be, but we have shown that one can learn much larger models reliably with much less memory than a naive implementation would allow. Moreover, our learning algorithms allow a flexible tradeoff between memory requirements and model complexity requirements that can be adapted online.

Intuitively, our method may be thought of as a two level clustering process. At the bottom level, data is clustered into clumps in order to limit time and space costs. At the top level, clumps are clustered to form topics in order to ensure good generalization performance.

Potential application areas of the techniques introduced here are manyfold. For instance, we can imagine learning topic models from very large text corpora or the world wide web to understand its structure and facilitate fast searching algorithms. Another exciting direction is to build a taxonomy of visual object categories from a continuous stream of video data captured by surveillance cameras.

3.7.1 Appendix

The following expressions for the Free Energy are used in the main text. Note that they are only valid after the updates for $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ have been performed.

$$\begin{aligned}\mathcal{F} = KB(\boldsymbol{\beta}) - \sum_k B_k(\mathbf{F}_k + \boldsymbol{\beta}) + \sum_{ks} D_s \log\left(\Gamma(\alpha_k)/\Gamma(\alpha_k + \frac{N_{ks}}{D_s})\right) \\ - \sum_s D_s \log\left(\Gamma(\alpha)/\Gamma(\alpha + \frac{N_s}{D_s})\right) + \sum_{ck} N_c q_{ck} \log q_{ck} \\ - \sum_k \left((a-1)\sum_k \log(\alpha_k) - b\sum_k \alpha_k\right) \\ -(K_{\max} - K)\left((a-1)\log\frac{a-1}{b} - (a-1)\right) \\ -K_{\max}\left(b\log(a) - \log\Gamma(a)\right)\end{aligned} \tag{3.29}$$

$$\begin{aligned}\mathcal{F}_c =& KB(\boldsymbol{\beta}) - \sum_k B_k(\left(\frac{T_{pts}}{\sum_c N_c}\right)\mathbf{F}_k + \boldsymbol{\beta}) \\ &+ \left(\frac{T_{docs}}{\sum_s D_s}\right)\sum_{ks} D_s \log\left(\Gamma(\alpha_k)/\Gamma(\alpha_k + \frac{N_{ks}}{D_s})\right) \\ &- \left(\frac{T_{docs}}{\sum_s D_s}\right)\sum_s D_s \log\left(\Gamma(\alpha)/\Gamma(\alpha + \frac{N_s}{D_s})\right) + \left(\frac{T_{pts}}{\sum_c N_c}\right)\sum_{ck} N_c q_{ck} \log q_{ck} \\ &- \sum_k \left((a-1)\sum_k \log(\alpha_k) - b\sum_k \alpha_k\right) \\ &- (K_{\max} - K)\left((a-1)\log\frac{a-1}{b} - (a-1)\right) \\ &- K_{\max}\left(b\log(a) - \log\Gamma(a)\right)\end{aligned} \tag{3.30}$$

Chapter 4

Budgeted Nonparametric Learning from Data Streams

4.1 Abstract

We consider the problem of extracting informative exemplars from a data stream. Examples of this problem include exemplar-based clustering and nonparametric inference such as Gaussian process regression on massive data sets. We show that these problems require maximization of a submodular function that captures the informativeness of a set of exemplars, over a data stream. We develop an efficient algorithm, StreamGreedy*, which is guaranteed to obtain a constant fraction of the value achieved by the optimal solution to this NP-hard optimization problem. We extensively evaluate our algorithm on large real-world data sets.*

4.2 Introduction

Modern machine learning is increasingly confronted with the challenge of very large data sets. The unprecedented growth in text, video, and image data demands techniques that can effectively learn from large amounts of data, while still remaining computationally tractable. *Streaming* algorithms [GZK05, DH00, GMM+03, COP03] represent an attractive approach to handling the data deluge. In this model the learning system has access to a small fraction of the data set at any point in time, and cannot necessarily control the order in which the examples are visited. This is

particularly useful when the data set is too large to fit in primary memory, or if it is generated in real time and predictions are needed in a timely fashion.

While computational tractability is critical, powerful methods are required in order to learn useful models of complex data. Nonparametric learning methods are promising because they can construct complex decision rules by allowing the data to 'speak for itself'. They may use complex similarity measures that capture domain knowledge while still providing more flexibility than parametric methods. However, nonparametric techniques are difficult to apply to large datasets because they typically associate a parameter with every data point, and thus depend on all the data. Therefore, most algorithms for nonparametric learning operate in batch mode. To overcome this difficulty, nonparametric learning methods may be approximated by specifying a budget: a fixed limit on the number of examples that are used to make predictions.

In this work, we develop a framework for budgeted nonparametric learning that can operate in a streaming data environment. In particular, we study sparse Gaussian process regression and exemplar-based clustering under complex, non-metric distance functions, which both meet the requirements of our framework. The unifying concept of our approach is *submodularity*, an intuitive diminishing returns property. When a nonparametric problem's objective function satisfies this property, we show that a simple algorithm, StreamGreedy, may be used to choose examples from a data stream. We use submodularity to prove strong theoretical guarantees for our algorithm. We demonstrate our approach with experiments involving sparse Gaussian process regression and large scale exemplar-based clustering of 1.5 million images.

4.3 Problem Statement

We consider the problem of extracting a subset $\mathcal{A} \subseteq \mathcal{V}$ of k representative items from a large data set $\mathcal{V}$ (which can, e.g., consist of vectors in $\mathbb{R}^d$ or other objects such as graphs, lists, etc.). Our goal is to maximize a set function F that quantifies the utility $F(\mathcal{A})$ of any possible subset $\mathcal{A} \subseteq \mathcal{V}$. We give examples of such utility functions in

Section 4.4. Intuitively, in the clustering example, $F(\mathcal{A})$ measures, e.g., the reduction in quantization error when selecting exemplars $\mathcal{A}$ as cluster centers. In Gaussian process (GP) regression, $F(\mathcal{A})$ measures the prediction performance when selecting the active set $\mathcal{A}$. As we show below, many utility functions, such as those arising in clustering and GP regression, satisfy *submodularity*, an intuitive diminishing returns property: Adding a cluster center helps more if we have selected few exemplars so far, and less if we have already selected many exemplars. Formally, a set function F is said to be *submodular*, if for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $s \in \mathcal{V} \setminus \mathcal{B}$ it holds that

$$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B}).$$

An additional natural assumption is that F is *monotonic*, i.e., $F(\mathcal{A}) \leq F(\mathcal{B})$ whenever $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$.

Since the data set $\mathcal{V}$ is large, it is not possible to store it in memory, and we hence can only access a small number of items at any given time t. Let $\mathcal{B}_1, \ldots, \mathcal{B}_T, \ldots$ be a sequence of subsets of $\mathcal{V}$, where $\mathcal{B}_t$ is the set of elements in $\mathcal{V}$ that are available to the algorithm at time t. Typically $|\mathcal{B}_t| = m \ll n = |\mathcal{V}|$. For example, hardware limitations may require us to read data from disk, one block $\mathcal{B}_t$ of data points at a time.

We only assume that there is a bound ρ, such that for each element $b \in \mathcal{V}$, if $b \notin \mathcal{B}_t \cup \cdots \cup \mathcal{B}_{t+\ell}$, then $\ell < \rho$, i.e., we have to wait at most ρ steps until b reappears. This assumption is satisfied, for example, if $\mathcal{B}_t$ is a sliding window over the data set (in which case $\rho = n$), or $\mathcal{V}$ is partitioned into blocks, and the $\mathcal{B}_t$ cycle through these blocks (in which case ρ is $n/(\min_i |\mathcal{B}_i|)$). Our goal is to select at each time t a subset $\mathcal{A}_t \subseteq \mathcal{A}_{t-1} \cup \mathcal{B}_t$, $|\mathcal{A}_t| \leq k$, in order to maximize $F(\mathcal{A}_T)$ after some number of iterations T. Thus, at each time t we are allowed to pick any combination of k items from both the previous selection $\mathcal{A}_{t-1}$ and the available items $\mathcal{B}_t$, and we would like to maximize the final value $F(\mathcal{A}_T)$.

Our streaming assumptions mirror those of [COP03], in that we assume a finite data set in which data items may be revisited although the order is not under our

control. For certain submodular objectives (F_V and F_C but not F_H, see Section 4.4) we require the additional assumption that we may access data items uniformly at random (see Section 4.5).

Note that even if $\mathcal{B}_1 = \cdots = \mathcal{B}_T = \mathcal{V}$, i.e., access to the entire data set is always available, the problem of choosing a set

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})$$

maximizing a submodular function F is an NP-hard optimization problem [Fei98]. Hence, we cannot expect to efficiently find the optimal solution in general. The setting where $\mathcal{B}_t \subsetneq \mathcal{V}$ is strictly more general and thus harder. In this paper, we will develop an efficient approximation algorithm with strong theoretical guarantees for this problem.

4.4 Examples of Online Budgeted Learning

In this section, we discuss concrete problem instances of the streaming budgeted learning problem, and the corresponding submodular objective functions F.

Active set selection in GPs. Gaussian processes have been widely used as a powerful tool for nonparametric regression [RW06, Cre91]. Formally, a Gaussian process (GP) is a joint probability distribution $P(\mathcal{X}_\mathcal{V})$ over a (possibly infinite) set of random variables $\mathcal{X}_\mathcal{V}$ indexed by a set $\mathcal{V}$, with the property that every finite subset $\mathcal{X}_\mathcal{A}$ for $\mathcal{A} = \{s_1, \ldots, s_k\}$, $\mathcal{A} \subseteq \mathcal{V}$ is distributed according to a multivariate normal distribution, $P(\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \mathcal{N}(\mathbf{x}_\mathcal{A}; \mu_\mathcal{A}, \Sigma_{\mathcal{A}\mathcal{A}})$, where $\mu_\mathcal{A} = (\mathcal{M}(s_1), \ldots, \mathcal{M}(s_k))$ is the prior mean and

$$\Sigma_{\mathcal{A}\mathcal{A}} = \begin{pmatrix} \mathcal{K}(s_1, s_1) & \ldots & \mathcal{K}(s_1, s_k) \\ \vdots & & \vdots \\ \mathcal{K}(s_k, s_1) & \ldots & \mathcal{K}(s_k, s_k) \end{pmatrix}$$

is the prior covariance, parameterized through the positive definite kernel function $\mathcal{K}$. In GP regression, each data point $s \in \mathcal{V}$ is interpreted as a random variable in a GP. Based on observations $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$ of a subset $\mathcal{A}$ of variables, the predictive distribution of a new data point $s \in \mathcal{V}$ is a normal distribution $P(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \mathcal{N}(\mu_{s|\mathcal{A}}; \sigma^2_{s|\mathcal{A}})$, where

$$\mu_{s|\mathcal{A}} = \mu_s + \Sigma_{s\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}(\mathbf{x}_\mathcal{A} - \mu_\mathcal{A}) \tag{4.4.1}$$

$$\sigma^2_{s|\mathcal{A}} = \sigma^2_s - \Sigma_{s\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}s}, \tag{4.4.2}$$

and $\Sigma_{s\mathcal{A}} = (\mathcal{K}(s, s_1), \ldots, \mathcal{K}(s, s_k))$ and $\Sigma_{\mathcal{A}s} = \Sigma^T_{s\mathcal{A}}$. Computing the predictive distributions according to (4.4.1) is expensive, as it requires 'inverting' (finding the Cholesky decomposition) of the kernel matrix $\Sigma_{\mathcal{A}\mathcal{A}}$, which, in general requires $\Theta(|\mathcal{A}|^3)$ floating point operations. Reducing this computational complexity (and thereby enabling GP methods for large data sets) has been subject of much research (see [RW06]).

Most approaches for efficient inference in GPs rely on choosing a small *active set* $\mathcal{A}$ of data points for making predictions. For example, the informative vector machine (IVM) uses the set $\mathcal{A}$ that maximizes the information gain

$$F_H(\mathcal{A}) = H(\mathcal{X}_\mathcal{V}) - H(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_\mathcal{A}), \tag{4.4.3}$$

or, equivalently, the entropy $H(\mathcal{X}_\mathcal{A})$ of the random variables associated with the selected data points $\mathcal{A}$. It can be shown, that this criterion is monotonic and submodular [See04]. While efficiently computable, the IVM criterion F_H only depends on the selected data points, and does not explicitly optimize the prediction error of the non-selected examples $\mathcal{V} \setminus \mathcal{A}$.

An alternative is to choose data points which minimize the prediction accuracy on the non-selected data: $\widehat{L}(\mathcal{A}) = \sum_{s \in \mathcal{V}\setminus\mathcal{A}}(\mathbf{x}_s - \mu_{s|\mathcal{A}})^2$. If the data points $\mathcal{V}$ are drawn from some distribution $P(s)$, then this criterion can be seen as a sample approximation

to the expected variance reduction,

$$\begin{aligned}\widehat{L}(\mathcal{A}) &\approx \int P(s) \int P(\mathbf{x}_s \mid \mathbf{x}_\mathcal{A})(\mathbf{x}_s - \mu_{s|\mathcal{A}})^2 ds d\mathbf{x}_s \\ &= \int P(s)\sigma^2_{s|\mathcal{A}} d\mathbf{x}_s = L(\mathcal{A}).\end{aligned}$$

It can be shown, that under certain assumptions on the kernel function, the *expected variance reduction*

$$F_V(\mathcal{A}) = L(\emptyset) - L(\mathcal{A}) \tag{4.4.4}$$

is a monotonic submodular function.

Exemplar-based clustering with complex distance functions on data streams. In exemplar clustering problems, the goal is to select a set of examples from the data set that are representative of the data set as a whole. Exemplar clustering is particularly relevant in cases where choosing cluster centers that are averages of training examples (as in the k-means algorithm) is inappropriate or impossible (see [DF07] for examples). The k-medoid [KR90] approach seeks to choose exemplars that minimize the average dissimilarity of the data items to their nearest exemplar:

$$L(\mathcal{A}) = \frac{1}{|\mathcal{V}|} \sum_{s \in \mathcal{V}} \min_{c \in \mathcal{A}} d(\mathbf{x}_s, \mathbf{x}_c). \tag{4.4.5}$$

This loss function can be transformed to a monotonic submodular utility function by introducing a *phantom exemplar* $\mathbf{x}_0$ which may not be removed from the active set, and defining the utility function

$$F_C(\mathcal{A}) = L(\{\mathbf{x}_0\}) - L(\mathcal{A} \cup \{\mathbf{x}_0\}). \tag{4.4.6}$$

This measures the decrease in the loss associated with the active set versus the loss associated with just the phantom exemplar, and maximizing this function is equivalent to minimizing (4.4.5). The dissimilarity function $d(\mathbf{x}, \mathbf{x}')$ need only be a positive function of $\mathbf{x}$ and $\mathbf{x}'$, making this approach potentially very powerful.

4.5 StreamGreedy for Budgeted Learning from Data Streams

If, at every time, full access to the entire data set $\mathcal{V}$ is available, a simple approach to selecting the subset $\mathcal{A}_T$ would be to start with the empty set, $\mathcal{A}_0 = \emptyset$, and, at iteration t, greedily select the element

$$s_t = \operatorname*{argmax}_{s \in \mathcal{V}} F(\mathcal{A}_{t-1} \cup \{s\}) \tag{4.5.1}$$

for $t \leq k$, and $\mathcal{A}_t = \mathcal{A}_{t-1}$ for $t > k$. Perhaps surprisingly, this simple greedy algorithm is guaranteed to obtain a near-optimal solution: [NWF78] prove that for the solution $\mathcal{A}_T$, for any $T \geq k$, obtained by the greedy algorithm it holds that $F(\mathcal{A}_T) \geq (1 - 1/e) \max_{|\mathcal{A}| \leq k} F(\mathcal{A})$, i.e., it achieves at least a constant fraction of $(1 - 1/e)$ of the optimal value. In fact, no efficient algorithms can provide better approximation guarantees unless P=NP [Fei98].

Unfortunately, the greedy selection rule (4.5.1) requires access to all elements of $\mathcal{V}$, and hence cannot be applied in the streaming setting. A natural extension to the streaming setting is the following algorithm: Initialize $\mathcal{A}_0 = \emptyset$. For $t \leq k$, set $\mathcal{A}_t \leftarrow \mathcal{A}_{t-1} \cup \{s_t\}$, where

$$s_t = \operatorname*{argmax}_{s \in \mathcal{B}_t} F(\mathcal{A}_{t-1} \cup \{s\}). \tag{4.5.2}$$

For $t > k$, let

$$(s', s) = \operatorname*{argmax}_{s' \in \mathcal{A}_{t-1}, s \in \mathcal{A}_{t-1} \cup \mathcal{B}_t} F(\mathcal{A}_{t-1} \setminus \{s'\} \cup \{s\}), \tag{4.5.3}$$

and set $\mathcal{A}_t = \mathcal{A}_{t-1} \setminus \{s'\} \cup \{s\}$, i.e., replace item s' by item s in order to greedily maximize the utility. Stop after no significant improvement (at least η for some small value $\eta > 0$) is observed after a specified number ρ of iterations. StreamGreedy is summarized in Algorithm 4.

Algorithm 4 StreamGreedy

Initialize active set $\mathcal{A}_0 = \emptyset$; Bound ρ on wait time
for $t = 1 : k$ **do**
  Set $s_t = \operatorname{argmax}_{s \in \mathcal{B}_t} F(\mathcal{A}_{t-1} \cup \{s\})$
  Set $\mathcal{A}_t \leftarrow \mathcal{A}_{t-1} \cup \{s_t\}$
end for
Set $NI = 0$
while $NI \leq \rho$ **do**
  Set $(s', s) = \operatorname{argmax}_{s' \in \mathcal{A}_{t-1}, s \in \mathcal{A}_{t-1} \cup \mathcal{B}_t} F(\mathcal{A}_{t-1} \setminus \{s'\} \cup \{s\})$
  Set $t \leftarrow t + 1$; $\mathcal{A}_t = \mathcal{A}_{t-1} \setminus \{s'\} \cup \{s\}$
  if $F(\mathcal{A}_t) > F(\mathcal{A}_{t-1}) + \eta$ **then**
    Set $NI = 0$
  else
    Set $NI = NI + 1$
  end if
end while

Dealing with limited access to the stream. So far, we have assumed that StreamGreedy can evaluate the objective function F for any candidate set $\mathcal{A}$. While the IVM objective $F_H(\mathcal{A})$ for active set selection in GPs (see Section 4.4) only requires access to the selected data points $\mathcal{A}$, evaluating the objectives F_C and F_V requires access to the entire data set $\mathcal{V}$. However, these objective functions share a key property: They additively decompose over the data set. Hence, they can be written in the form

$$F(\mathcal{A}) = \frac{1}{|\mathcal{V}|} \sum_{s \in \mathcal{V}} f(\mathcal{A}, \mathbf{x}_s)$$

for suitable function f such that $f(\cdot, \mathbf{x}_s)$ is submodular for each input $\mathbf{x}_s$. If we assume that data points $\mathbf{x}_s$ are generated i.i.d. from a distribution and f is a measurable function of $\mathbf{x}_s$, then $f(\mathcal{A}, \mathbf{x}_s)$ are themselves a series of i.i.d. outcomes of a random variable. Moreover, the range of random variables $f(\mathcal{A}, \mathbf{x}_s)$ is bounded by some constant B (for clustering, B is the diameter of the data set; for GP regression, B is the maximum prior marginal variance). We can construct a sample approximation $\widehat{F}(\mathcal{A}) = \frac{1}{|\mathcal{W}|} \sum_{s \in \mathcal{W}} f(\mathcal{A}, \mathbf{x}_s)$ by choosing a validation set $\mathcal{W}$ uniformly at random from the stream $\mathcal{V}$.

The following corollary of Hoeffding's inequality adapted from [SMS99] bounds

the deviation between $\widehat{F}(\mathcal{A})$ and $F(\mathcal{A})$:

Corollary 1 ([SMS99]). *Let* $c = \frac{B^2 \log(\frac{2}{\delta})}{2|\mathcal{V}|\varepsilon^2}$ *and* $\delta > 0$. *Then, with probability* $1 - \delta$ *for* $|\mathcal{W}| = \frac{c}{1+c}|\mathcal{V}|$*:*

$$\left| \frac{1}{|\mathcal{W}|} \widehat{F}(\mathcal{A}) - \frac{1}{|\mathcal{V}|} F(\mathcal{A}) \right| < \varepsilon$$

.

The result relates the level of approximation to the fraction of the data set that is needed for validation. As the number of elements in the stream $|\mathcal{V}|$ increases, smaller fractions are needed to reach a given accuracy. Because this result holds for any (bounded) data distribution, it is usually pessimistic; in practice, smaller validation sets often suffice. Furthermore, this sample-based approximation only requires a constant amount of memory: When $\mathbf{x}_s$ arrives from the stream, $f(\mathcal{A}, \mathbf{x}_s)$ may be added to a sufficient statistic and $\mathbf{x}_s$ itself may be discarded.

4.6 Theoretical Analysis

Clustering-consistent objectives. For clarity of notation, we will consider the setting where $\mathcal{B}_t = \{b_t\}$ contains only a single element $b_t \in \mathcal{V}$. The results generalize to sets $\mathcal{B}_t$ containing more elements.

We first show that for an interesting class of submodular functions, the algorithm actually converges to the optimal solution. Suppose, the data set $\mathcal{V}$ can be partitioned into a set of clusters, i.e., $\mathcal{V} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_L$, where $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$. We call a monotonic submodular function F *clustering-consistent* for a particular clustering $\mathcal{C}_1, \ldots, \mathcal{C}_L$, if the following conditions hold:

1. $F(\mathcal{A}) = \sum_{\ell=1}^{L} F(\mathcal{A} \cap \mathcal{C}_\ell)$, i.e., F decomposes additively across clusters.

2. Whenever for two sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ such that $\mathcal{B} = \mathcal{A} \cup \{s\} \setminus \{s'\}$, $s \in \mathcal{C}_i$, $s' \in \mathcal{C}_j$, $i \neq j$ it holds that if $|\mathcal{A} \cap \mathcal{C}_j| > 1$ and $\mathcal{A} \cap \mathcal{C}_i = \emptyset$, then $F(\mathcal{A}) \leq F(\mathcal{B})$.

Intuitively, a submodular function F is clustering-consistent, if it is always preferable

to select a representative from a new cluster than having two representatives of the same cluster.

Proposition 2. *Suppose F is clustering-consistent for $\mathcal{V}$ and $k \leq L$. Then, for $T = 2\rho$ it holds for all sets $\mathcal{A}_t$, $t \geq T$ returned by* StreamGreedy *(for $\eta = 0$) that*

$$F(\mathcal{A}_t) = \max_{|\mathcal{A}| \leq k} F(\mathcal{A}).$$

The proofs can be found in this chapter's Appendix. Thus, for clustering-consistent objectives F, if the data set really consists of L clusters, and we use StreamGreedy to select a set of $k \leq L$ exemplars, then StreamGreedy converges to the optimal solution after at most two passes through the data set $\mathcal{V}$.

A key question is which classes of objective functions are clustering-consistent. In the following, suppose that the elements in $\mathcal{V}$ are endowed with a metric d. The following proposition gives interesting examples:

Proposition 3. *Suppose $\mathcal{V} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_L$, $|\mathcal{C}_i| < \alpha |\mathcal{C}_j|$ for all i, j. Further suppose that*

$$\max_i \operatorname{diam}(\mathcal{C}_i) < \beta \min_{i,j} d(\mathcal{C}_i, \mathcal{C}_j)$$

for suitable constants α and β, where $d(\mathcal{C}_i, \mathcal{C}_j) = \min_{r \in \mathcal{C}_i, s \in \mathcal{C}_j} d(r, s)$ and $\operatorname{diam}(\mathcal{C}_i) = \max_{r,s \in \mathcal{C}_i} d(r, s)$. Then the following objectives from Section 4.4 are clustering-consistent with $\mathcal{V} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_L$:

- *The clustering objective F_C, whenever $\max_{\mathbf{x} \in \mathcal{C}_i} d(\mathbf{x}, \mathbf{x}_0) \leq \min_j d(\mathcal{C}_i, \mathcal{C}_j)$ for all i, j, where $\mathbf{x}_0$ is the phantom exemplar.*
- *The entropy F_H and variance reduction[1] F_V for Gaussian process regression with squared exponential kernel functions with appropriate bandwidth σ^2, and where d is the Euclidean metric in $\mathbb{R}^d$.*

Intuitively, Propositions 2 and 3 suggests that in situations where the data actually exhibits a well-separated, balanced clustering structure, and we are interested in

[1] under the condition of conditional suppressor-freeness [DK08]

selecting a number of exemplars k consistent with the number of clusters L in the data, we expect StreamGreedy to perform near-optimally.

General submodular objectives. However, the assumptions made by Propositions 2 and 3 are fairly strong, and likely violated by the existence of outliers, overlapping and imbalanced clusters, etc. Furthermore, when using criteria such as F_C and F_V (Section 4.4), it is not possible to evaluate $F(\mathcal{A})$ exactly, but only up to additive error ε. Perhaps surprisingly, even in such more challenging settings, the algorithm is still guaranteed to converge to a near-optimal solution:

Theorem 4. *Let $\eta > 0$. Suppose F is monotonic submodular on $\mathcal{V}$, and we have access to a function $\widehat{F}$ such that for all $\mathcal{A} \subseteq \mathcal{V}$, $|\mathcal{A}| \leq 2k$ it holds that $|\widehat{F}(\mathcal{A}) - F(\mathcal{A})| \leq \varepsilon$. Furthermore suppose F is bounded by B. Then, for $T = \rho B/\eta$ it holds for all sets $\mathcal{A}_t$, $t \geq T$ selected by* StreamGreedy *applied to $\widehat{F}$ that*

$$F(\mathcal{A}_t) \geq \frac{1}{2} \max_{|\mathcal{A}| \leq k} F(\mathcal{A}) - k(\varepsilon + \eta).$$

Thus, e.g., in the case where $b_t = s_{t \bmod n}$, i.e., if StreamGreedy sequentially cycles through the data set $\mathcal{V}$, at most B/η passes (typically it will stop far earlier) through the data set will suffice to produce a solution that obtains almost half the optimal value. The proof relies on properties of the pairwise exchange heuristic for submodular functions [NWF78]. See this chapter's Appendix for details.

4.7 Experimental Results

Exemplar-based streaming clustering. Our exemplar-based clustering experiments involve StreamGreedy applied to the clustering utility F_C (Eq. 4.4.6) with $d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2$. The implementation can be made efficient by exploiting the fact that only a subset of the validation points (see Section 4.5) change cluster membership for each candidate swap. We have also implemented an adaptive stopping rule that is useful when determining an appropriate size of the validation set. Please see

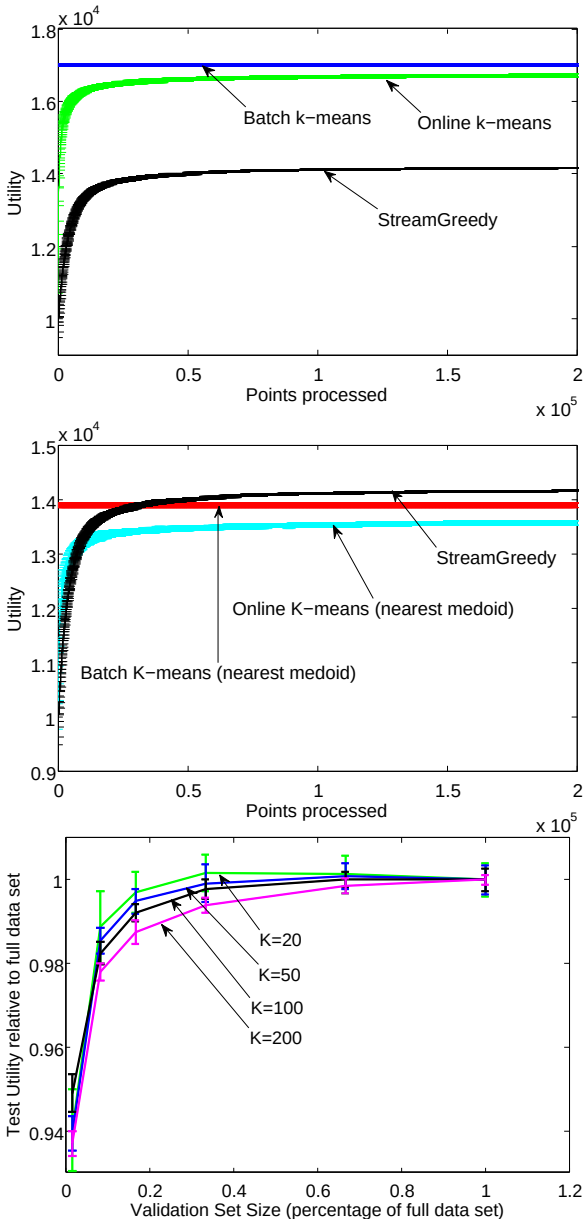


Figure 4.1: Top and Center: Convergence rates on MNIST data set. The y-axis represents the clustering utility evaluated on the training set. The x-axis shows the number of data items processed by StreamGreedy and online k-means. K-means' unconstrained centers yield better quantization performance. When k-means' centers are replaced with the nearest training set example, the advantage disappears (center). Bottom: Test performance versus validation set size. It is possible to obtain good generalization performance even using relatively small validation sets. The validation set size is varied along the x-axis. The y-axis shows test utility divided by the test utility achieved with the entire data set used for validation. As K increases, more validation data is needed to achieve full performance.

this chapter's Appendix II for details.

Our first set of experiments uses MNIST handwritten digits with 60,000 training images and 10,000 test images.[2] The MNIST digits were preprocessed as follows: The 28 by 28 pixel images are initially represented as 784 dimensional vectors, and the mean of the training image vectors was subtracted from each image; then the resulting vectors are normalized to unit norm. PCA was performed on the normalized training vectors and the first 50 principal components coefficients were used to form feature vectors. The same normalization procedure was performed on the test images and their dimensionality was also reduced using the training PCA basis.

Figure 4.1 compares the performance of our approach against batch k-means and online k-means [Das09] with the number of exemplars set to K = 100. We chose the origin as the *phantom exemplar* in this experiment, since this yielded better overall quantization performance than choosing a random exemplar. To unambiguously assess convergence speed we use the entire training set of 60,000 points as the validation set. We assess convergence by plotting (4.4.6) against the number of swap candidates ($\sum_{t=1}^{T} |\mathcal{B}_t|$) considered. We find that our algorithm converges to a solution after examining nearly the same number of data points as online k-means, and is near its final value after a single pass through the training data. Similar convergence was observed for smaller validation sizes. The top plot in Figure 4.1 shows that k-means performs better in terms of quantization loss. This is probably because StreamGreedy must choose exemplar centers from the training data, while k-means center locations are unconstrained. When the k-means' centers are replaced with the nearest training example (center plot), the advantage disappears. The bottom plot in Figure 4.1 examines the impact of validation set size on quantization performance on the held out test set, measured as test set utility ((4.4.6) where $\mathcal{V}$ is the test set). It is possible to obtain good generalization performance even when using a small validation set. The y-axis indicates test performance relative to the performance attained with the full data set at the specified value of K (1.0 indicates equal performance, values less than one indicate worse performance than the full set), and the x-axis is plotted as

[2]MNIST was downloaded from `http://yann.lecun.com/exdb/mnist/`.

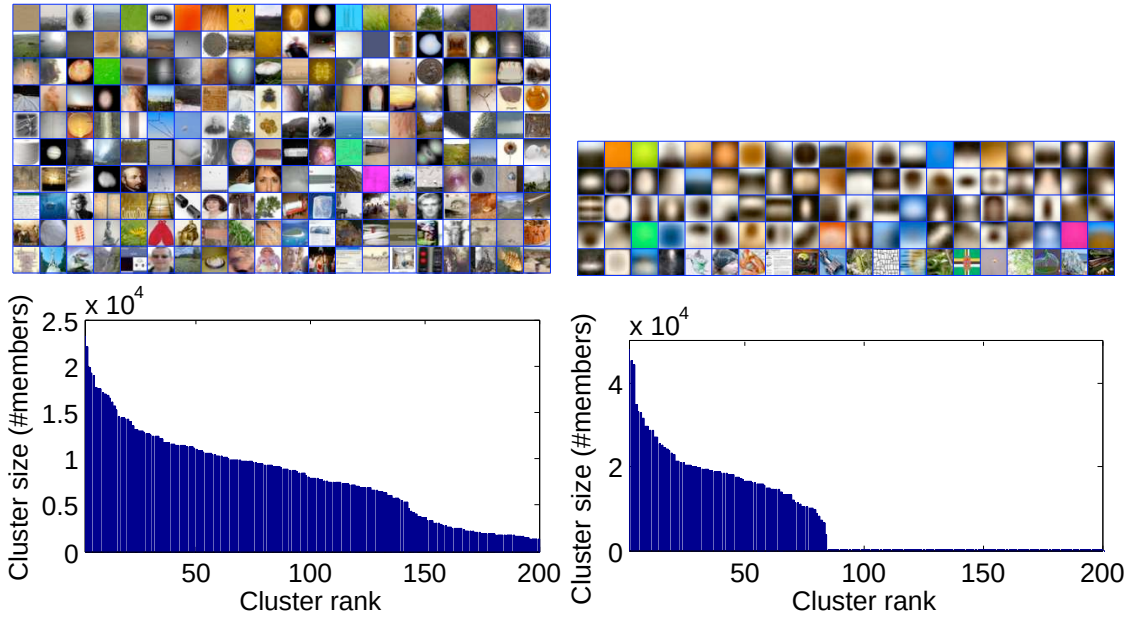


Figure 4.2: Tiny Image data set. Top Left: Cluster exemplars discovered by StreamGreedy, sorted according to descending size. Top Right: Cluster centers from online kmeans (singleton clusters omitted). Bottom Left: Cluster sizes (number of members) for our algorithm. Bottom Right: Cluster sizes for online k-means. Online k-means finds a poor local minima with many of the 200 clusters containing only a single member.

the relative size of the validation set versus the full set. We find that as the number of centers K increases, a larger fraction of the data set is needed to approach the performance with the full set. This appears to be because as K increases, the numerical differences between $F_C(\mathcal{A}_{t-1} \setminus \{s'\} \cup \{s\})$ for alternative candidate swaps (s, s') decrease, and more samples are needed in order to stably rank the swap alternatives.

Our second set of experiments involves approximately 1.5 million *Tiny Images*[3] (described in [TFF08]), and is designed to test our algorithm on a large scale dataset. Each image in the dataset was downloaded by Torralba et al. from an Internet search engine and is associated with an English noun query term. The 32 by 32 RGB pixel images are represented as 3,072 dimensional vectors. Following [TFF08], we subtract from each vector its mean value (average of all components), then normalize it to unit norm. No dimensionality reduction is performed. We generate a random center

[3] `http://people.csail.mit.edu/torralba/tinyimages/`

Figure 4.3: Examples from Tiny Image cluster 26. Left: 100 examples nearest to exemplar 26. Right: 100 randomly sampled images from cluster 26

to serve as the *phantom exemplar* for this experiment, since we find that this leads to qualitatively more interesting clusters than using the origin[4].

Figure 4.2 (left) shows $K = 200$ exemplars discovered by our algorithm. Clusters are organized primarily according to non-semantic visual characterstics such as color and basic shape owing to the simple sum of squared differences similarity measure employed (Figure 4.3). We set the validation size to one-fifth of the data set. This was determined by examining the stability of $\text{argmax}_{s' \in \mathcal{A}_{t-1}, s \in \mathcal{A}_{t-1} \cup \mathcal{B}_t} F_C(\mathcal{A}_{t-1} \setminus \{s'\} \cup \{s\})$ as validation data was progressively added to the sums in F_C, which tends to stabilize well before this amount of data is considered. The algorithm was halted after 600 iterations (each considering $|\mathcal{B}_t| = 1,000$ candidate centers). This was determined based on inspection of the utility function, which converged before a single pass through the data. We compare against the online k-means algorithm with 200 centers initialized to randomly chosen images, and run through a single pass over the data. We find that online k-means converges to a suboptimal solution in which many of the clusters are empty or contain only a single member (see Figure 4.2.)

[4]We find that a random phantom exemplar is unlikely to be chosen as a prototype, while one near the origin is the prototype for a significant fraction of the data.

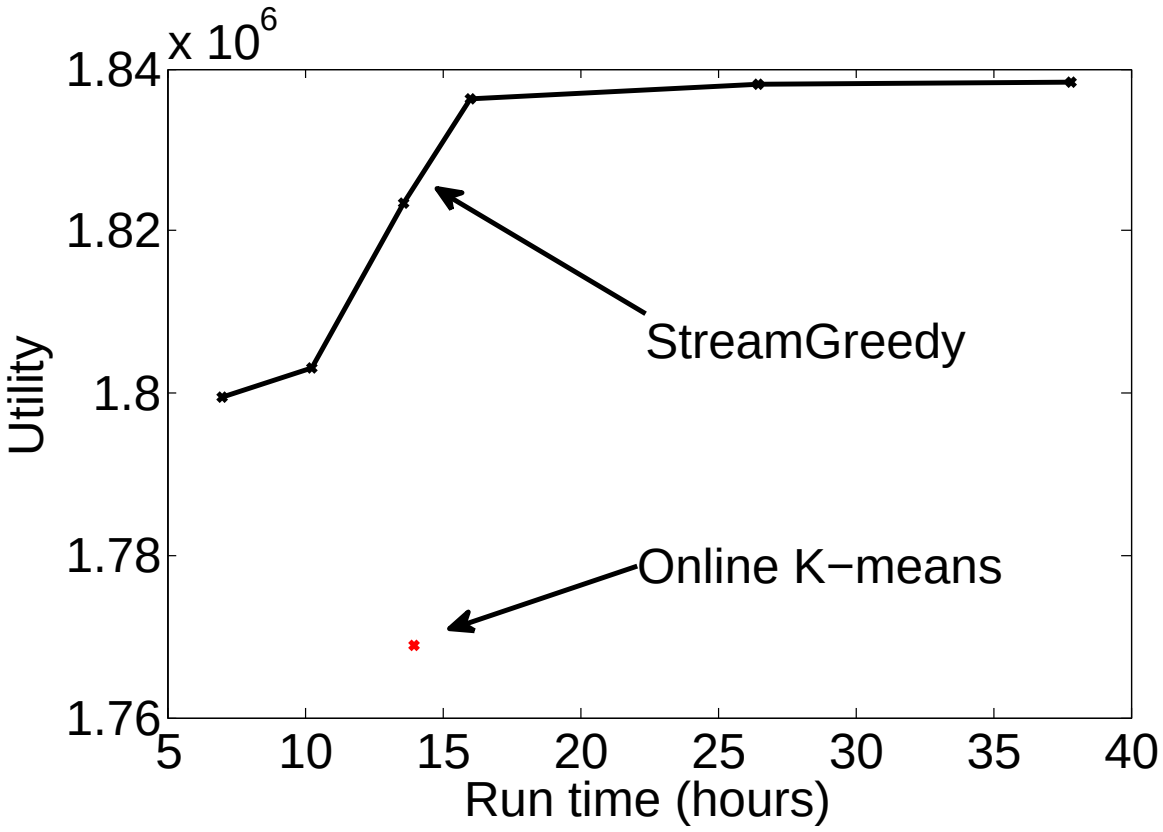


Figure 4.4: Utility score versus run time on the Tiny Images data set. We find that the performance of StreamGreedy tends to saturate with run time, which is determined by varying the number of items presented by the stream at each iteration ($|\mathcal{B}_t|$) and the validation set size ($|\mathcal{W}|$). StreamGreedy outperforms online K-means at all run time settings.

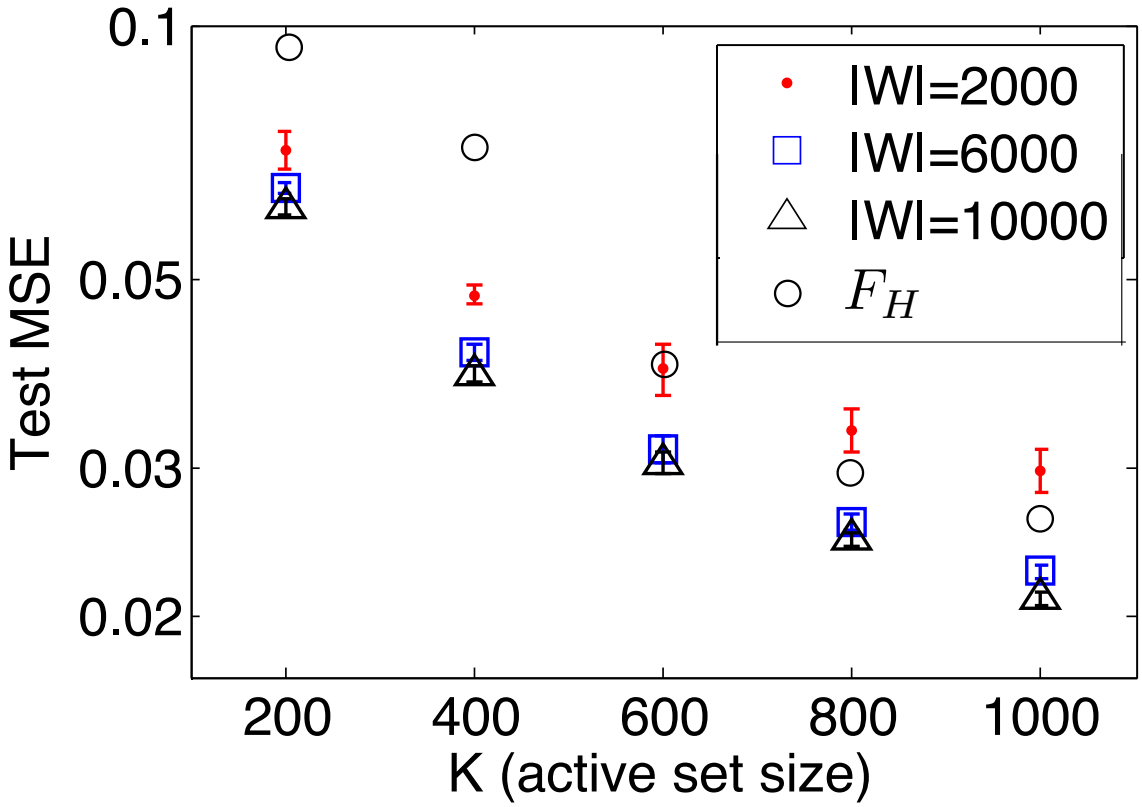


Figure 4.5: Gaussian process regression. The y-axis is the mean squared prediction error evaluated over the held out test set. The x-axis is the size of the active set. Performance is plotted for the F_V objective at varying validation set sizes ($|\mathcal{W}|$). We find that when the active set size is small the performance is similar across validation size settings. As the active set size increases, performance is enhanced as the validation set size increases. The F_H objective (circles) does not require a validation set, and the performance gap between F_V decreases as the active set size increases.

In Figure 4.4 we assess the tradeoff between run time and performance by varying the parameter $|\mathcal{B}_t| = \{500, 1000, 2000\}$ and the validation set size as $\{10\%, 20\%, 40\%\}$ of the data set. The number of centers and iterations are fixed at 200 and 600, respectively. Our Matlab StreamGreedy implementation was run on a quad-core Intel Xeon server. Performance for each parameter setting is visualized as a point in the test utility versus run time plane, and only the Pareto optimal points are displayed for clarity. Online k-means is also shown for comparison. We find that StreamGreedy achieves higher utility at less running time, and a clear saturation in performance occurs as run time increases.

Online active set selection for GP regression. Our Gaussian process regression experiments involve specialization of StreamGreedy for the objective function F_V in Section 4.4. The implementation can be made more efficient by using Cholesky factorization on the covariance matrix combined with rank one updates and downdates. Please see this chapter's Appendix for details. We used the KIN40K dataset[5] which consists of 9 attributes generated by a robotic arm simulator. We divide the dataset into 10,000 training and 30,000 test instances. We follow the preprocessing steps outlined by [SWL03] in order to compare our approach to the results in that study. We used the squared exponential kernel with automatic relevance determination (ARD) weights and learn the hyperparameters using marginal likelihood maximization [RW06] on a subset of 2,000 training points, again following [SWL03].

Figure 4.5 shows the mean squared error predictive performance $\frac{1}{2}\sum_s (y_s - \mu_s)^2$ on the test set as a function of the size of the active set. Comparing our results to the experiments of [SWL03], we find that our approach outperforms the info-gain criterion for active set size $K = \{200, 400, 600\}$ at all values of the validation set size $|\mathcal{W}| = \{2000, 6000, 10000\}$. At values $K = \{800, 1000\}$ our approach outperforms info-gain for $|\mathcal{W}| = \{6000, 10000\}$. Our performance matches [SB00] at $K = \{200, 400\}$ but slightly underperforms their approach at larger values of K. We find that even for $|\mathcal{W}| = 2,000$, the algorithm is able to gain predictive ability by choosing more active

[5]Downloaded from http://ida.first.fraunhofer.de/ anton/data.html

examples from the data stream. The performance gap between $|\mathcal{W}| = 6,000$ and $|\mathcal{W}| = 10,000$ is quite small.

4.8 Related Work

Specialization of StreamGreedy to the clustering objective F_C (4.4.6) yields an algorithm which is similar to the Partitioning Around Medoids (PAM, [KR90]) algorithm for k-medoids, and related algorithms CLARA [KR90] and CLARANS [NH02]. Like our approach, these algorithms are based on repeatedly exchanging centers for non-center data points if the swap improves the objective function. Unlike our approach, however, no performance guarantees are known for these approaches. PAM requires access to the entire data set, and every data point is exhaustively examined at each iteration, leading to an approach unsuitable for large databases. CLARA runs PAM repeatedly on subsamples of the data set, but then makes use of the entire dataset when comparing the results of each PAM run. Like our algorithm, CLARANS evaluates a random subset of candidate centers at each iteration, but then makes use of the entire data set to evaluate candidate swaps. Our approach takes advantage of the i.i.d. concentration behavior of the clustering objective in order to eliminate the need for accessing the entire data set, while still yielding a performance guarantee. [DH01] exploit the concentration behavior of the (non-exemplar) k-means objective in a similar way. While there exist online algorithms for k-medoids with strong theoretical guarantees [COP03], these algorithms require the distance function d to be a metric, and the memory to grow (logarithmically) in $|\mathcal{V}|$. In contrast, our approach uses arbitrary dissimilarity functions and the memory requirements are independent of the data set size.

Specialization of StreamGreedy to sparse GP inference is an example of the *subset of datapoints* class of sparse Gaussian process approximations [RW06], in which the GP predictive distribution is conditioned on only the datapoints in the active set. [SWL03] also use a *subset of datapoints* approach that makes use of a submodular [See04] utility function (the entropy of the Gaussian distribution of each site in

the active set). This approach is computationally cheaper than ours in that the evaluation criterion does not require a validation set, but depends only on the current active set. Seeger et al.'s approach also fits the framework proposed by this paper, and our approach could be used to optimize this objective over data streams. [SB00] use a *subset of regressors* approach. Their criterion for greedy selection of regressors has the same complexity as our approach if we use the entire data set for validation. Our approach is cheaper when we make use of a limited validation set. [CO02] develop an approach for online sparse GP inference based on *projected process* approximation that also involves swapping candidate examples into an active set, but without performance guarantees. See [RW06] for a survey of other methods for sparse Gaussian process approximation.

StreamGreedy's structure is similar to the algorithm by [WBB05] for online learning of kernel perceptron classifiers, in that both approaches make use of a fixed budget of training examples (the active set) that are selected by evaluating a loss function defined over a limited validation set.

[NWF78] analyzed the greedy algorithm and a pairwise exchange algorithm for maximizing submodular functions. As argued in Section 4.5, these algorithms do not apply to the streaming setting. [SG08] develop an online algorithm for maximizing a sequence of submodular functions over a fixed set (that needs to be accessed every iteration). Our approach, in contrast, maximizes a single submodular function on a sequence of sets, using bounded memory.

4.9 Conclusions

We have developed a theoretical framework for extracting informative exemplars from data streams that led to StreamGreedy, an effective algorithm with strong theoretical guarantees. We have shown that this framework can be successfully specialized to exemplar-based problems and nonparametric regression with Gaussian processes. In the case of clustering, our experiments demonstrate that our approach is capable of discovering meaningful clusters in large high-dimensional data sets, while remaining

computationally tractable. Our sparse Gaussian process regression algorithm is competitive with respect to other approaches and is capable of operating in a streaming data environment. Future work involves discovering other machine learning problems that fit the framework (including classification) and exploring alternative ways to approximately evaluate submodular functions without full access to a large data set.

4.10 Appendix I: Proofs

Proof of Proposition 2. Suppose F is clustering-consistent for clustering $\mathcal{C}_1, \dots, \mathcal{C}_L$. We prove Proposition 2 in two steps:

Let T_1 be such that at least one element $b \in \mathcal{C}_i$ has been encountered for each cluster $\mathcal{C}_i$. Then, for the solution $\mathcal{A}_{T_1}$ it holds that $|\mathcal{C}_i \cap \mathcal{A}_{T_1}| \leq 1$ for each i, i.e., $\mathcal{A}_{T_1}$ contains at most one representative of each cluster: Let t_i be the smallest index such that $b_{t_i} \in \mathcal{C}_i$ (i.e., the first iteration where a representative of cluster i appears in the stream). W.l.o.g., assume that $t_1 < t_2 < \cdots < t_L \leq T_1$. For any set $\mathcal{A} \subseteq \mathcal{V}$, let

$$r(\mathcal{A}) = |\{i : \mathcal{C}_i \cap \mathcal{A} \neq \emptyset\}|$$

denote the number of clusters from which at least one representative has been selected. By definition of clustering-consistency, it can be seen that for the sequence of sets $\mathcal{A}_1, \dots, \mathcal{A}_T$ chosen by StreamGreedy it holds that $r(\mathcal{A}_1) \leq \cdots \leq r(\mathcal{A}_T)$, i.e., it is never preferable to remove a single representative s of cluster $\mathcal{C}_i$ in order to have multiple representatives of some cluster $\mathcal{C}_j$. Moreover, it can be seen that

$$r(\mathcal{A}_{t_\ell}) = \min\{\ell, k\}.$$

Note that $T_1 \leq \rho$.

For the second step, note that for each $t \geq T_1$, it holds that $r(\mathcal{A}_t) = k$, i.e., k clusters will be represented, i.e., no set $\mathcal{A}_t$ will contain more than one exemplar from

any cluster i. Let

$$s_i^* = \operatorname*{argmax}_{s \in \mathcal{C}_i} F(\{s\})$$

be the (w.l.o.g. unique) highest scoring representative of cluster i. Assume, w.l.o.g., that $F(\{s_1^*\}) \geq F(\{s_2^*\}) \geq \cdots \geq F(\{s_L^*\})$. Due to the first condition of cluster-consistency (additive decomposition), it can be seen that

$$F(\{s_1^*, \ldots, s_k^*\}) = \max_{|\mathcal{A}| \leq k} F(\mathcal{A}).$$

Let $t_i^* \geq T_1$ be the smallest integer such that $b_{t_i^*} = s_i^*$ where the element s_i^* appears in the stream. It can be seen that, for all $\ell \leq k$ and for all $t \geq t_\ell^*$ it holds that $s_\ell^* \in \mathcal{A}_t$, hence at time $T = t_k$ it must hold that $F(\mathcal{A}_T) = \max_{|\mathcal{A}| \leq k} F(\mathcal{A})$. Note that $T \leq 2\rho$. □

Proof of Proposition 3. First consider the clustering objective $F = F_C$. Let

$$L_i(\mathcal{A}) = \sum_{s \in \mathcal{C}_i} \min_{c \in \mathcal{A} \cup \{\mathbf{x}_0\}} d(\mathbf{x}_s, \mathbf{x}_c)$$

be the loss associated with cluster $\mathcal{C}_i$. Let $\mathcal{A} \subseteq \mathcal{C}_i$. Note that if $s \in \mathcal{C}_j$ for $j \neq i$, then $L_i(\mathcal{A} \cup \{s\}) = L_i(\mathcal{A})$, since $d(\mathbf{x}_{s'}, \mathbf{x}_0) \leq d(\mathbf{x}_{s'}, \mathbf{x}_s)$ for all $s' \in \mathcal{C}_i$. Hence, for any $\mathcal{A} \subseteq \mathcal{V}$, $F(\mathcal{A}) = \sum_{\ell=1}^{L} F(\mathcal{A} \cap \mathcal{C}_\ell)$. Now suppose $\mathcal{A} \subseteq \mathcal{C}_i$ and $s \in \mathcal{C}_i \setminus \mathcal{A}$. Then

$$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \leq |\mathcal{C}_i| \operatorname{diam}(\mathcal{C}_i).$$

On the other hand, if $s \in \mathcal{C}_j$, then

$$F(\{j\}) \geq |\mathcal{C}_j|(d(\mathbf{x}_0, \mathcal{C}_j) - \operatorname{diam}(\mathcal{C}_j)).$$

Hence, choosing

$$\alpha = \frac{\min_i d(\mathbf{x}_0, \mathcal{C}_i) - \max_j \operatorname{diam}(\mathcal{C}_j)}{\min_i \operatorname{diam}(\mathcal{C}_i)}$$

suffices to prove cluster-consistency of F_C. Choosing

$$\beta = \frac{\min_i d(\mathbf{x}_0, \mathcal{C}_i)}{\min_{i,j} d(\mathcal{C}_i, \mathcal{C}_j)}$$

suffices to ensure that $\alpha > 0$.

Now let us consider active set selection on GP regression. Under the squared exponential kernel with bandwidth h,

$$\mathcal{K}(s, s') = \eta^2 \exp(-d(s, s')^2/h^2),$$

for any $\varepsilon > 0$, there is a constant c, such that for two sets $\mathcal{A}, \mathcal{B}$ with $d(\mathcal{A}, \mathcal{B}) > ch$, it holds that $|H(\mathcal{X}_\mathcal{A}, \mathcal{X}_\mathcal{B}) - H(\mathcal{X}_\mathcal{A}) - H(\mathcal{X}_\mathcal{B})| < \varepsilon$, and similarly $|\sigma_s^2 - \sigma_{s|\mathcal{A}}^2| \leq \varepsilon$, whenever $s \in \mathcal{B}$. This proves the additive decomposition property (up to arbitrarily small error ε; Proposition 2 can be generalized to accommodate this arbitrarily small error). Let $L_i(\mathcal{A}) = \sum_{s \in \mathcal{C}_i} [\sigma_s^2 - \sigma_{s|\mathcal{A}}^2]$. Now, there exists a constant c' such that if $\operatorname{diam}(\mathcal{C}_i) < c'h$ then for any $s \in \mathcal{C}_i$ and $\gamma < 1$ it holds that $L_i(\{s\}) < \gamma |\mathcal{C}_i| \eta^2$, and thus

$$F(\{s\}) > (1 - \gamma)|\mathcal{C}_i|.$$

Similarly, for any $\mathcal{A} \subseteq \mathcal{C}_i$ and $s \in \mathcal{C}_i \setminus \mathcal{A}$,

$$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) < \gamma |\mathcal{C}_i|,$$

proving cluster-consistency for appropriate choice of α. A similar reasoning can be used to prove cluster-consistency of the IVM objective F_H. □

Proof of Theorem 4. StreamGreedy can be interpreted as an instance of the pairwise exchange heuristic for submodular functions, which iteratively replaces a selected element by a non-selected element until no further improvement in score is possible, with the difference that the choice of candidate elements for replacement is determined by the data stream. The proof of Theorem 4 is thus analogous to the analysis of the pairwise exchange heuristic for submodular functions by [NWF78],

exploiting the key insight that the ordering in which pairwise exchanges are performed is immaterial for the performance guarantee of the pairwise exchange heuristic. The proof below also accommodates the fact that F is only evaluated up to small additive error ε (by means of $\widehat{F}$), and improvement of at least η is required for each exchange. Let T be index such that $\mathcal{A}_t = \mathcal{A}_T$ for all sets $\mathcal{A}_t$, $t \geq T$, chosen by StreamGreedy. Such a T must exist, since $F(\mathcal{A}_{t+1}) \geq F(\mathcal{A}_t)$ for all t, and $\mathcal{A}_{t+1} \neq \mathcal{A}_t$ only if $F(\mathcal{A}_{t+1}) \geq F(\mathcal{A}_t) + \eta$, and $F(\mathcal{V})$ is bounded. Construct an ordering $s_1, \ldots, s_k$ such that $s_i \in \operatorname{argmax}_{s \in \mathcal{A}_T} F(\{s_1, \ldots, s_{i-1}, s\})$. Also let $\mathcal{A}^* = \{r_1, \ldots, r_k\}$ such that $F(\mathcal{A}^*) = \max_{|\mathcal{A}| \leq k} F(\mathcal{A})$. Let $\mathcal{S} = \{s_1, \ldots, s_{k-1}\}$, and $\delta_i = F(\{s_1, \ldots, s_i\}) - F(\{s_1, \ldots, s_{i-1}\})$. Note that $\delta_i \leq \delta_{i-1}$, due to submodularity and the fact that $s_1, \ldots, s_k$ are in greedy order. Now, due to submodularity and monotonicity of F it holds that

$$
\begin{aligned}
F(\mathcal{A}^*) &\leq F(\mathcal{A}^* \cup \mathcal{S}) \\
&\leq F(\mathcal{S}) + \sum_{i=1}^{k} \left[F(\mathcal{S} \cup \{r_i\}) - F(\mathcal{S})\right] \\
&\leq F(\mathcal{S}) + k\,(\delta_k + \varepsilon + \eta) && (4.10.1) \\
&\leq F(\mathcal{S}) + \sum_{i=1}^{k} \delta_i && (4.10.2) \\
&= F(\mathcal{S}) + F(\mathcal{S} \cup \{s_k\}) \\
&\leq 2F(\mathcal{A}_T)
\end{aligned}
$$

where inequality (4.10.1) follows from the fact that StreamGreedy did not replace s_k by any element r_i from the optimal solution $\mathcal{A}^*$. Note that after ρ iterations, $F(\mathcal{A}_t)$ must increase by at least η, or StreamGreedy will stop. Hence, $T \leq \rho F(\mathcal{V})/\eta \leq \rho B/\eta$. □

4.11 Appendix II: Implementation Details

4.11.1 Clustering

When determining the swap to perform at iteration $t > K$, we maintain the following quantities from iteration $t-1$:

- The distance of each validation point $i \in \mathcal{W}$ to its cluster exemplar:

$$m_i = \min_{c \in A_{t-1} \cup \{\mathbf{x}_0\}} d(\mathbf{x}_i, \mathbf{x}_c). \tag{4.11.1}$$

- The identity of each validation point's exemplar:

$$z_i = \arg \min_{c \in A_{t-1} \cup \{\mathbf{x}_0\}} d(\mathbf{x}_i, \mathbf{x}_c). \tag{4.11.2}$$

- The distance of each validation point to its second nearest exemplar

$$o_i = \min_{c \in A_{t-1} \cup \{\mathbf{x}_0\} \setminus z_i} d(\mathbf{x}_i, \mathbf{x}_c). \tag{4.11.3}$$

We then compute the dissimilarity of each candidate in $\mathcal{B}_t$ to the points in the validation set $\mathcal{W}$ which requires $O(|\mathcal{B}_t||\mathcal{W}|\text{cost}\{d\})$ operations (where cost$\{d\}$ is the cost associated with computing the dissimilarity $d(\mathbf{x}, \mathbf{x}')$). We then score each of the $K|B_t|$ potential swaps (indexed by $s \in \mathcal{B}_t$ and $s' \in \mathcal{A}_{t-1}$) by computing

$$\begin{aligned} &L(\mathcal{A}_{t-1} \setminus \{s'\} \cup \{s, \mathbf{x}_0\}) \\ &\quad = L(\mathcal{A}_{t-1} \setminus \{s'\} \cup \{s, \mathbf{x}_0\}) - L(\mathcal{A}_{t-1} \cup \{s, \mathbf{x}_0\}) \\ &\quad + L(\mathcal{A}_{t-1} \cup \{s, \mathbf{x}_0\}) - L(\mathcal{A}_{t-1} \cup \{\mathbf{x}_0\}) \\ &\quad + L(\mathcal{A}_{t-1} \cup \{\mathbf{x}_0\}). \end{aligned}$$

This can be done in $O(|\mathcal{W}|)$ operations since the decrease in loss due to adding center s

$$L(\mathcal{A}_{t-1} \cup \{s, \mathbf{x}_0\}) - L(\mathcal{A}_{t-1} \cup \{\mathbf{x}_0\}) = \sum_{i:d(\mathbf{x}_i,\mathbf{x}_s)<m_i} d(\mathbf{x}_i, \mathbf{x}_s) - m_i$$

and the increase in loss associated with removing center s'

$$L(\mathcal{A}_{t-1} \setminus \{s'\} \cup \{s, \mathbf{x}_0\}) - L(\mathcal{A}_{t-1} \cup \{s, \mathbf{x}_0\}) = \sum_{i:z_i=s' \wedge [d(\mathbf{x}_i,\mathbf{x}_{s'})<d(\mathbf{x}_i,\mathbf{x}_s)]} o_i - m_i$$

require $O(|\mathcal{W}|)$. The third term $L(\mathcal{A}_{t-1} \cup \{\mathbf{x}_0\}) = \sum_{i \in \mathcal{W}} m_i$ doesn't depend on s or s'. The total cost for iteration t is $O(K|\mathcal{B}_t||\mathcal{W}| + |\mathcal{B}_t||\mathcal{W}|\text{cost}\{d\})$.

4.11.2 GP Regression

At each iteration t we retain from iteration $t-1$ the Cholesky decomposition of $\Sigma_{\mathcal{A}_{t-1},\mathcal{A}_{t-1}} = R_{t-1}^T R_{t-1}$, where R_{t-1} is an upper right triangular matrix, as well as the output of the kernel function $\mathcal{K}$ evaluated between points in $\mathcal{W}$ and $\mathcal{A}_{t-1}$. We compute $\mathcal{K}$ between each member of $\mathcal{B}_t$ and $\mathcal{W}$ as well as between $\mathcal{B}_t$ and $\mathcal{A}_{t-1}$ in $O((|\mathcal{W}| + K)|\mathcal{B}_t|\text{cost}(\mathcal{K}))$.

For each candidate swap indexed by $s \in \mathcal{B}_t$ and $s' \in \mathcal{A}_{t-1}$, we compute $R_{t-1}^{(s,s')}$ which is the Cholesky decomposition of $\Sigma_{\mathcal{A}_{t-1}\setminus s' \cup \{s\}, \mathcal{A}_{t-1}\setminus s' \cup \{s\}}$ with a downdate of element s' and update of element s performed in $O(K^2)$ operations. The prediction weight vector

$$\Sigma^{-1}_{\mathcal{A}_{t-1}\setminus s' \cup \{s\}, \mathcal{A}_{t-1}\setminus s' \cup \{s\}} \mathbf{x}_{\mathcal{A}_{t-1}\setminus s' \cup \{s\}}$$

can be computed in $O(K^2)$ operations using two forward substitutions, which corresponds to matrix right division in Matlab:

$$R_{t-1}^{(s,s')} \backslash ([R_{t-1}^{(s,s')}]^T \backslash \mathbf{x}_{\mathcal{A}_{t-1}\setminus s' \cup \{s\}}).$$

The candidate swaps are scored according to $\sum_{i \in \mathcal{W}} (\mathbf{x}_i - \mu_{i|\mathcal{A}_{t-1} \setminus s' \cup \{s\}})^2$ in $O(K|\mathcal{W}|)$.

The total cost for iteration t is $O((|\mathcal{W}|+K)|\mathcal{B}_t|\text{cost}(\mathcal{K})+K^3|\mathcal{B}_t|+K^2|\mathcal{B}_t||\mathcal{W}|)$. We are exploring ways to reduce this cost that involve identifying and evaluating only a fraction of the $K|\mathcal{B}_t|$ possible swaps, while still maintaining convergence guarantees.

4.11.3 Adaptive Stopping Rule

We have implemented an adaptive stopping rule based on updating a sufficient statistic $\widehat{F}(\mathcal{A}_{t-1} \cup \{s\} \setminus \{s'\}) = \sum_{i \in \mathcal{W}} f(\mathcal{A}_{t-1} \cup \{s\} \setminus \{s'\}, \mathbf{x}_i)$ for each swap candidate (indexed by $s \in \mathcal{B}_t$ and $s' \in \mathcal{A}_{t-1}$). Each time a data point $\mathbf{x}_i$ arrives from the stream, $f(\mathcal{A}_{t-1} \cup \{s\} \setminus \{s'\}, \mathbf{x}_i)$ is added to its corresponding sufficient statistic. We define an algorithm failure probablity $\hat{\delta}$, and use Lemma 1 with $\delta = \frac{\hat{\delta}}{A}$ where A is the maximum number of times the bound will be used during the course of the algorithm. This establishes confidence bands $\varepsilon_{s,s'}$ around each statistic depending on the number of samples summarized so far by the sufficient statistics, as well as confidence band $\varepsilon_{\mathcal{A}_{t-1}}$ on $\widehat{F}(\mathcal{A}_{t-1})$ (the current utility). We halt when one of two conditions are met:

- There exists a swap (s, s') such that $\widehat{F}(\mathcal{A}_{t-1} \cup \{s\} \setminus \{s'\}) - \varepsilon_{s,s'} > \widehat{F}(\mathcal{A}_{t-1}) + \varepsilon_{\mathcal{A}_{t-1}}$. We then perform swap (s, s') and move on to the next iteration $t+1$.
- For all swaps (s, s'), $\widehat{F}(\mathcal{A}_{t-1} \cup \{s\} \setminus \{s'\}) + \varepsilon_{s,s'} < \widehat{F}(\mathcal{A}_{t-1}) - \varepsilon_{\mathcal{A}_{t-1}}$. No swap is performed and we move on to the next iteration $t+1$.

This setup is similar to Hoeffding Racing [MM94]. We have experimented with this rule on the Tiny Images data set. We find that in the early iterations, it can cut down the number of validation samples used by a factor between 3 and 10. However, as the algorithm proceeds, the difference in utility between swap candidates becomes smaller and eventually the entire data set is used. We observe that this approach is pessimistic: $\arg\max_{s,s'} \widehat{F}(\mathcal{A}_{t-1} \cup \{s\} \setminus \{s'\})$ stablizes with many fewer samples than required by the stopping rule.

Part II

Discriminative Clustering

Chapter 5

Discriminative Clustering by Regularized Information Maximization

5.1 Abstract

Is there a principled way to learn a probabilistic discriminative classifier from an unlabeled data set? We present a framework that simultaneously clusters the data and trains a discriminative classifier. We call it Regularized Information Maximization *(RIM). RIM optimizes an intuitive information-theoretic objective function which balances class separation, class balance, and classifier complexity. The approach can flexibly incorporate different likelihood functions, express prior assumptions about the relative size of different classes and incorporate partial labels for semi-supervised learning. In particular, we instantiate the framework to unsupervised, multi-class kernelized logistic regression. Our empirical evaluation indicates that RIM outperforms existing methods on several real data sets, and demonstrates that RIM is an effective model selection method.*

5.2 Introduction

Clustering algorithms group data items into categories without requiring human supervision or definition of categories. They are often the first tool used when explor-

ing new data. A great number of clustering principles have been proposed, most of which can be described as either *generative* or *discriminative* in nature. Generative clustering algorithms provide constructive definitions of categories in terms of their geometric properties in a feature space or as statistical processes for generating data. Examples include k-means and Gaussian mixture model clustering. In order for generative clustering to be practical, restrictive assumptions must be made about the underlying category definitions.

Rather than modeling categories explicitly, discriminative clustering techniques represent the boundaries or distinctions between categories. Fewer assumptions about the nature of categories are made, making these methods powerful and flexible in real-world applications. Spectral graph partitioning [NJW01] and maximum margin clustering [XS05] are example discriminative clustering methods. A disadvantage of existing discriminative approaches is that they lack a probabilistic foundation, making them potentially unsuitable in applications that require reasoning under uncertainty or in data exploration.

We propose a principled probabilistic approach to discriminative clustering, by formalizing the problem as unsupervised learning of a conditional probabilistic model. We generalize the work of Grandvalet and Bengio [GB04] and Bridle et al. [BHM92] in order to learn probabilistic classifiers that are appropriate for multi-class discriminative clustering, as explained in Section 5.3. We identify two fundamental, competing quantities, class balance and class separation, and develop an information theoretic objective function which trades off these quantities. Our approach corresponds to maximizing mutual information between the empirical distribution on the inputs and the induced label distribution, regularized by a complexity penalty. Thus, we call our approach Regularized Information Maximization (RIM).

In summary, our contribution is RIM, a probabilistic framework for discriminative clustering with a number of attractive properties. Thanks to its probabilistic formulation, RIM is flexible: it is compatible with diverse likelihood functions and allows specification of prior assumptions about expected class proportions. We show how our approach leads to an efficient, scalable optimization procedure that also pro-

vides a means of automatic model selection (determination of the number of clusters). RIM is easily extended to semi-supervised classification. Finally, we show that RIM performs better than competing approaches on several real-world data sets.

5.3 Regularized Information Maximization

Suppose we are given an unlabeled dataset of N feature vectors (datapoints) $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{iD})^T \in \mathbb{R}^D$ are D-dimensional vectors with components x_{id}. Our goal is to learn a conditional model $p(y|\mathbf{x}, \mathbf{W})$ with parameters $\mathbf{W}$ which predicts a distribution over label values $y \in \{1, \ldots, K\}$ given an input vector $\mathbf{x}$. Our approach is to construct a functional $F(p(y|\mathbf{x}, \mathbf{W}); \mathbf{X}, \lambda)$ which evaluates the suitability of $p(y|\mathbf{x}, \mathbf{W})$ as a discriminative clustering model. We then use standard discriminative classifiers such as logistic regression for $p(y|\mathbf{x}, \mathbf{W})$, and maximize the resulting function $F(\mathbf{W}; \mathbf{X}, \lambda)$ over the parameters $\mathbf{W}$. λ is an additional tuning parameter that is fixed during optimization.

We are guided by three principles when constructing $F(p(y|\mathbf{x}, \mathbf{W}); \mathbf{X}, \lambda)$. The first is that the discriminative model's decision boundaries should not be located in regions of the input space that are densely populated with datapoints. This is often termed the *cluster assumption* [CZ04], and also corresponds to the idea that datapoints should be classified with large margin. Grandvalet & Bengio [GB04] show that a conditional entropy term $-\frac{1}{N}\sum_i H\{p(y|\mathbf{x}_i, \mathbf{W})\}$ very effectively captures the cluster assumption when training probabilistic classifiers with partial labels. However, in the case of fully unsupervised learning this term alone is not enough to ensure sensible solutions, because conditional entropy may be reduced by simply removing decision boundaries and unlabeled categories tend to be removed. We illustrate this in Figure 5.1 (left) with an example using the multilogit regression classifier as the conditional model $p(y|\mathbf{x}, \mathbf{W})$, which we will develop in Section 5.4.

In order to avoid degenerate solutions, we incorporate the notion of class balance: we prefer configurations in which category labels are assigned evenly across the

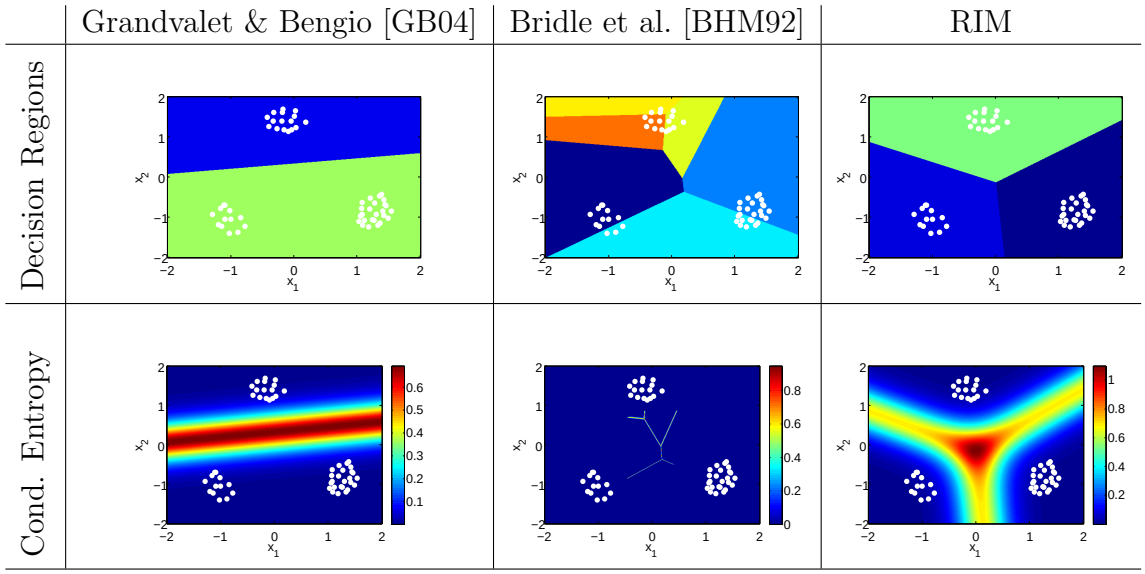


Figure 5.1: Example unsupervised multilogit regression solutions on a simple dataset with three clusters. The top and bottom rows show the category label $\arg\max_y p(y|\mathbf{x}, \mathbf{W})$ and conditional entropy $H\{p(y|\mathbf{x}, \mathbf{W})\}$ at each point $\mathbf{x}$, respectively. We find that both class balance and regularization terms are necessary to learn unsupervised classifiers suitable for multi-class clustering.

dataset. We define the empirical label distribution

$$\hat{p}(y; \mathbf{W}) = \int \hat{p}(\mathbf{x}) p(y|\mathbf{x}, \mathbf{W}) d\mathbf{x} = \frac{1}{N} \sum_i p(y|\mathbf{x}_i, \mathbf{W}),$$

which is an estimate of the marginal distribution of y. A natural way to encode our preference towards class balance is to use the entropy $H\{\hat{p}(y; \mathbf{W})\}$, because it is maximized when the labels are uniformly distributed. Combining the two terms, we arrive at

$$I_{\mathbf{W}}\{y; \mathbf{x}\} = H\{\hat{p}(y; \mathbf{W})\} - \frac{1}{N} \sum_i H\{p(y|\mathbf{x}_i, \mathbf{W})\} \tag{5.3.1}$$

which is the empirical estimate of the mutual information between $\mathbf{x}$ and y under the conditional model $p(y|\mathbf{x}, \mathbf{W})$.

Bridle et al. [BHM92] were the first to propose maximizing $I_{\mathbf{W}}\{y; \mathbf{x}\}$ in order to learn probabilistic classifiers without supervision. However, they note that $I_{\mathbf{W}}\{y; \mathbf{x}\}$

may be trivially maximized by a conditional model that classifies each data point $\mathbf{x}_i$ into its own category y_i, and that classifiers trained with this objective tend to fragment the data into a large number of categories, see Figure 5.1 (center). We therefore introduce a regularizing term $R(\mathbf{W};\lambda)$ whose form will depend on the specific choice of $p(y|\mathbf{x},\mathbf{W})$. This term penalizes conditional models with complex decision boundaries in order to yield sensible clustering solutions. Our objective function is

$$F(\mathbf{W};\mathbf{X},\lambda) = I_{\mathbf{W}}\{y;\mathbf{x}\} - R(\mathbf{W};\lambda) \tag{5.3.2}$$

and we therefore refer to our approach as *Regularized Information Maximization* (RIM), see Figure 5.1 (right). While we motivated this objective with notions of class balance and seperation, our approach may be interpreted as learning a conditional distribution for y that preserves information from the data set, subject to a complexity penalty.

5.4 Example Application: Unsupervised Multilogit Regression

The RIM framework is flexible in the choice of $p(y \mid \mathbf{x};\mathbf{W})$ and $R(\mathbf{W};\lambda)$. As an example instantiation, we here choose multiclass logistic regression as the conditional model.

Specifically, if K is the maximum number of classes, we choose

$$p(y=k|\mathbf{x},\mathbf{W}) \propto \exp(\mathbf{w}_k^T\mathbf{x} + b_k) \quad \text{and} \quad R(\mathbf{W};\lambda) = \lambda \sum_k \mathbf{w}_k^T\mathbf{w}_k, \tag{5.4.1}$$

where the set of parameters $\mathbf{W} = \{\mathbf{w}_1,\ldots,\mathbf{w}_K;b_1,\ldots,b_K\}$ consists of weight vectors $\mathbf{w}_k$ and bias values b_k for each class k. Each weight vector $\mathbf{w}_k \in \mathbb{R}^D$ is D-dimensional with components w_{kd}. The regularizer is the squared L_2 norm of the weight vectors, and may be interpreted as an isotropic normal distribution prior on the weights $\mathbf{W}$. The bias terms are not penalized.

In order to optimize Eq. 5.3.2 specialized with Eqs. 5.4.1, we require the gradients of the objective function. For clarity, we define $p_{ki} \equiv p(y = k|\mathbf{x}_i, \mathbf{W})$, and $\hat{p}_k \equiv \hat{p}(y = k; \mathbf{W})$. The partial derivatives are

$$\frac{\partial F}{\partial w_{kd}} = \frac{1}{N} \sum_{ic} \frac{\partial p_{ci}}{\partial w_{kd}} \log \frac{p_{ci}}{\hat{p}_c} - 2\lambda w_{kd} \quad \text{and} \quad \frac{\partial F}{\partial b_k} = \frac{1}{N} \sum_{ic} \frac{\partial p_{ci}}{\partial b_k} \log \frac{p_{ci}}{\hat{p}_c}. \tag{5.4.2}$$

Naive computation of the gradient requires $O(NK^2D)$, since there are $K(D + 1)$ parameters and each derivative requires a sum over NK terms. However, the form of the conditional probability derivatives for multi-logit regression are:

$$\frac{\partial p_{ci}}{\partial w_{kd}} = (\delta_{kc} - p_{ci}) p_{ki} x_{id} \quad \text{and} \quad \frac{\partial p_{ci}}{\partial b_k} = (\delta_{kc} - p_{ci}) p_{ki},$$

where δ_{kc} is equal to one when indices k and c are equal, and zero otherwise. When these expressions are substituted into Eq. 5.4.2, we find the following expressions:

$$\begin{aligned} \frac{\partial F}{\partial w_{kd}} &= \frac{1}{N} \sum_i x_{id} p_{ki} \Big(\log \frac{p_{ki}}{\hat{p}_k} - \sum_c p_{ci} \log \frac{p_{ci}}{\hat{p}_c} \Big) - 2\lambda w_{kd} \\ \frac{\partial F}{\partial b_k} &= \frac{1}{N} \sum_i p_{ki} \Big(\log \frac{p_{ki}}{\hat{p}_k} - \sum_c p_{ci} \log \frac{p_{ci}}{\hat{p}_c} \Big). \end{aligned} \tag{5.4.3}$$

Computing the gradient requires only $O(NKD)$ operations since the terms $\sum_c p_{ci} \log \frac{p_{ci}}{\hat{p}_c}$ may be computed once and reused in each partial derivative expression.

The above gradients are used in the L-BFGS [LN89] quasi-Newton optimization algorithm[1]. We find empirically that the optimization usually converges within a few hundred iterations. When specialized to multilogit regression, the objective function $F(\mathbf{W}; \mathbf{x}, \lambda)$ is non-concave. Therefore the algorithm can only be guaranteed to halt at locally optimal stationary points of F. In Section 5.4.1, we explain how we can obtain an initialization that is robust against local optima.

[1]We used Mark Schmidt's implementation at `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`.

5.4.1 Model Selection

Setting the derivatives (Eq. 5.4.3) equal to zero yields the following condition at stationary points of F:

$$\mathbf{w}_k = \sum_i \alpha'_{ki} \mathbf{x}_i \tag{5.4.4}$$

where we have defined

$$\alpha'_{ki} \equiv \frac{1}{2\lambda N} p_{ki} \Big(\log \frac{p_{ki}}{\hat{p}_k} - \sum_c p_{ci} \log \frac{p_{ci}}{\hat{p}_c} \Big). \tag{5.4.5}$$

The L_2 regularizing function $R(\mathbf{W}; \lambda)$ in Eq. 5.4.1 is additively composed of penalty terms associated with each category: $\mathbf{w}_k^T \mathbf{w}_k = \sum_{ij} \alpha'_{ki} \alpha'_{kj} \mathbf{x}_i^T \mathbf{x}_j$. It is instructive to observe the limiting behavior of the penalty term $\mathbf{w}_k^T \mathbf{w}_k$ when datapoints are not assigned to category k; that is, when $\hat{p}_k = \frac{1}{N} \sum_i p_{ki} \to 0$. This implies that $p_{ki} \to 0$ for all i, and therefore $\alpha'_{ki} \to 0$ for all i. Finally, $\mathbf{w}_k^T \mathbf{w}_k = \sum_{ij} \alpha'_{ki} \alpha'_{kj} \mathbf{x}_i^T \mathbf{x}_j \to 0$. This means that the regularizing function does not penalize unpopulated categories.

We find empirically that when we initialize with a large number of category weights $\mathbf{w}_k$, many decay away depending on the value of λ. Typically as λ increases, fewer categories are discovered. This may be viewed as model selection (automatic determination of the number of categories) since the regularizing function and parameter λ may be interpreted as a form of prior on the weight parameters. The bias terms b_k are unpenalized and are adjusted during optimization to drive the class probablities $\hat{p}_k$ arbitrarily close to zero for unpopulated classes. This is illustrated in Figure 5.2.

This behavior suggests an effective initialization procedure for our algorithm. We first oversegment the data into a large number of clusters (using k-means or other suitable algorithm) and train a *supervised* multi-logit classifier using these cluster labels. (This initial classifier may be trained with a small number of L-BFGS iterations since it only serves as a starting point.) We then use this classifier as the starting point for our RIM algorithm and optimize with different values of λ in order to obtain solutions with different numbers of clusters.

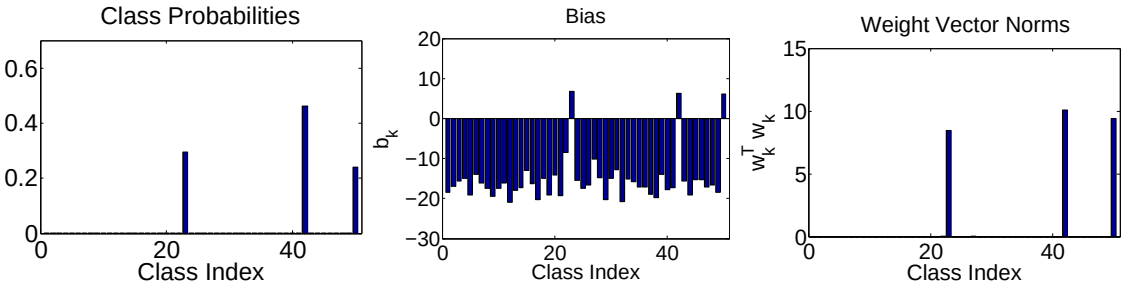


Figure 5.2: Demonstration of model selection on the toy problem from Figure 5.1. The algorithm is initialized with 50 category weight vectors $\mathbf{w}_k$. Upon convergence, only three of the categories are populated with data examples. The negative bias terms of the unpopulated categories drive the unpopulated class probabilities $\hat{p}_k$ towards zero. The corresponding weight vectors $\mathbf{w}_k$ have norms near zero.

5.5 Example Application: Unsupervised Kernel Multilogit Regression

The stationary conditions have another interesting consequence. Eq. 5.4.4 indicates that at stationary points, the weights are located in the span of the input datapoints. We use this insight as justification to define explicit coefficients α_{ki} and enforce the constraint $\mathbf{w}_k = \sum_i \alpha_{ki}\mathbf{x}_i$ during optimization. Substituting this equation into the multilogit regression conditional likelihood allows replacement of all inner products $\mathbf{w}_k^T\mathbf{x}$ with $\sum_i \alpha_{ki}K(\mathbf{x}_i, \mathbf{x})$, where K is a positive definite kernel function that evaluates the inner product $\mathbf{x}_i^T\mathbf{x}$. The conditional model now has the form

$$p(y = k|\mathbf{x}, \alpha, \mathbf{b}) \propto \exp\Big(\sum_i \alpha_{ki}K(\mathbf{x}_i, \mathbf{x}) + b_k\Big).$$

Substituting the constraint into the regularizing function $\sum_k \mathbf{w}_k^T\mathbf{w}_k$ yields a natural replacement of $\mathbf{w}_k^T\mathbf{w}_k$ by the Reproducing Hilbert Space (RKHS) norm of the function $\sum_i \alpha_{ki}K(\mathbf{x}_i, \cdot)$:

$$R(\alpha) = \sum_k \sum_{ij} \alpha_{ki}\alpha_{kj}K(\mathbf{x}_i, \mathbf{x}_j). \tag{5.5.1}$$

We use the L-BFGS algorithm to optimize the kernelized algorithm over the coefficients α_{ki} and biases b_k. The partial derivatives for the kernel coefficients are

$$\frac{\partial F}{\partial \alpha_{kj}} = \frac{1}{N}\sum_i K(\mathbf{x}_j, \mathbf{x}_i) p_{ki}\Big(\log\frac{p_{ki}}{\hat{p}_k} - \sum_c p_{ci}\log\frac{p_{ci}}{\hat{p}_c}\Big) - 2\lambda\sum_i \alpha_{ki}K(\mathbf{x}_j, \mathbf{x}_i)$$

and the derivatives for the biases are unchanged. The gradient of the kernelized algorithm requires $O(KN^2)$ to compute. Kernelized unsupervised multilogit regression exhibits the same model selection behavior as the linear algorithm.

5.6 Extensions

We now discuss how RIM can be extended to semi-supervised classification, and to encode prior assumptions about class proportions.

5.6.1 Semi-Supervised Classification

In semi-supervised classification, we assume that there are unlabeled examples $\mathbf{X}^U = \{\mathbf{x}_1^U, \cdots, \mathbf{x}_N^U\}$ as well as labeled examples $\mathbf{X}^L = \{\mathbf{x}_1^L, \cdots, \mathbf{x}_M^L\}$ with labels $Y = \{y_1, \cdots, y_M\}$.

We again use mutual information $I_{\mathbf{W}}\{y; \mathbf{x}\}$ (Eq. 5.3.1) to define the relationship between unlabeled points and the model parameters, but we incorporate an additional parameter τ which will define the tradeoff between labeled and unlabeled examples. The conditional likelihood is incorporated for labeled examples to yield the semi-supervised objective:

$$S(\mathbf{W}; \tau, \lambda) = \tau I_{\mathbf{W}}\{y; \mathbf{x}\} - R(\mathbf{W}; \lambda) + \sum_i \log p(y_i | \mathbf{x}_i^L, \mathbf{W}).$$

The gradient is computed and again used in the L-BFGS algorithm in order to optimize this combined objective. Our approach is related to the objective in [GB04], which does not contain the class balance term $H(\hat{p}(y; \mathbf{W}))$.

5.6.2 Encoding Prior Beliefs about the Label Distribution

So far, we have motivated our choice for the objective function F through the notion of class balance. However, in many classification tasks, different classes have different number of members. In the following, we show how RIM allows flexible expression of prior assumptions about non-uniform class label proportions.

First, note that the following basic identity holds

$$H\{\hat{p}(y;\mathbf{W})\} = \log(K) - KL\{\hat{p}(y;\mathbf{W})||U\} \tag{5.6.1}$$

where U is the uniform distribution over the set of labels $\{1,\cdots,K\}$. Substituting the identity, then dropping the constant $\log(K)$ yields another interpretation of the objective

$$F(\mathbf{W};\mathbf{X},\lambda) = -\frac{1}{N}\sum_i H\{p(y|\mathbf{x}_i,\mathbf{W})\} - KL\{\hat{p}(y;\mathbf{W})||U\} - R(\mathbf{W};\lambda). \tag{5.6.2}$$

The term $-KL\{\hat{p}(y;\mathbf{W})||U\}$ is maximized when the average label distribution is uniform. We can capture prior beliefs about the average label distribution by substituting a reference distribution $D(y;\gamma)$ in place of U (γ is a parameter that may be fixed or optimized during learning). [JMJ99] also use relative entropy as a means of enforcing prior beliefs, although not with respect to class distributions in multi-class classification problems.

This construction may be used in a clustering task in which we believe that the cluster sizes obey a power law distribution as, for example, considered by [Teh06] who use the Pitman-Yor process for nonparametric language modeling. Simple manipulation yields the following objective:

$$F(\mathbf{W};\mathbf{X},\lambda,\gamma) = I_{\mathbf{W}}\{\mathbf{x};y\} - H\{\hat{p}(y;\mathbf{W})||D(y;\gamma)\} - R(\mathbf{W};\lambda)$$

where $H\{\hat{p}(y;\mathbf{W})||D(y;\gamma)\}$ is the cross entropy $-\sum_k \hat{p}(y=k;\mathbf{W})\log D(y=k;\gamma)$. We therefore find that label distribution priors may be incorporated using an addi-

tional cross entropy regularization term.

5.7 Experiments

We empirically evaluate our RIM approach on several real data sets, in both fully unsupervised and semisupervised configurations.

5.7.1 Unsupervised Learning

Kernelized RIM is initialized according to the procedure outlined in Section 5.4.1, and run until L-BFGS converges. Unlabeled examples are then clustered according to $\arg\max_k p(y = k|\mathbf{x}, \mathbf{W})$. We compare RIM against the spectral clustering (SC) algorithm of [NJW01], the fast maximum margin clustering (MMC) algorithm of [ZTK07], and kernelized k-means [STC04]. MMC is a binary clustering algorithm. We use the recursive scheme outlined by [ZTK07] to extend the approach to multiple categories. The MMC algorithm requires an initial clustering estimate for initialization, and we use SC to provide this.

We evaluate unsupervised clustering performance in terms of how well the discovered clusters reflect known ground truth labels of the dataset. We report the Adjusted Rand Index (ARI) [HA85] between an inferred clustering and the ground truth categories. ARI has a maximum value of 1 when two clusterings are identical. We evaluated a number of other measures for comparing clusterings to ground truth including mutual information, normalized mutual information [SG02], and cluster impurity [CWK05]. We found that the relative rankings of the algorithms were the same as indicated by ARI.

We evaluate the performance of each algorithm while varying the number of clusters that are discovered, and we plot ARI for each setting. For SC and k-means the number of clusters is given as an input parameter. MMC is evaluated at $\{2, 4, 8, \cdots\}$ clusters (powers of two, due to the recursive scheme.) For RIM, we sweep the regularization parameter λ and allow the algorithm to discover the final number of clusters.

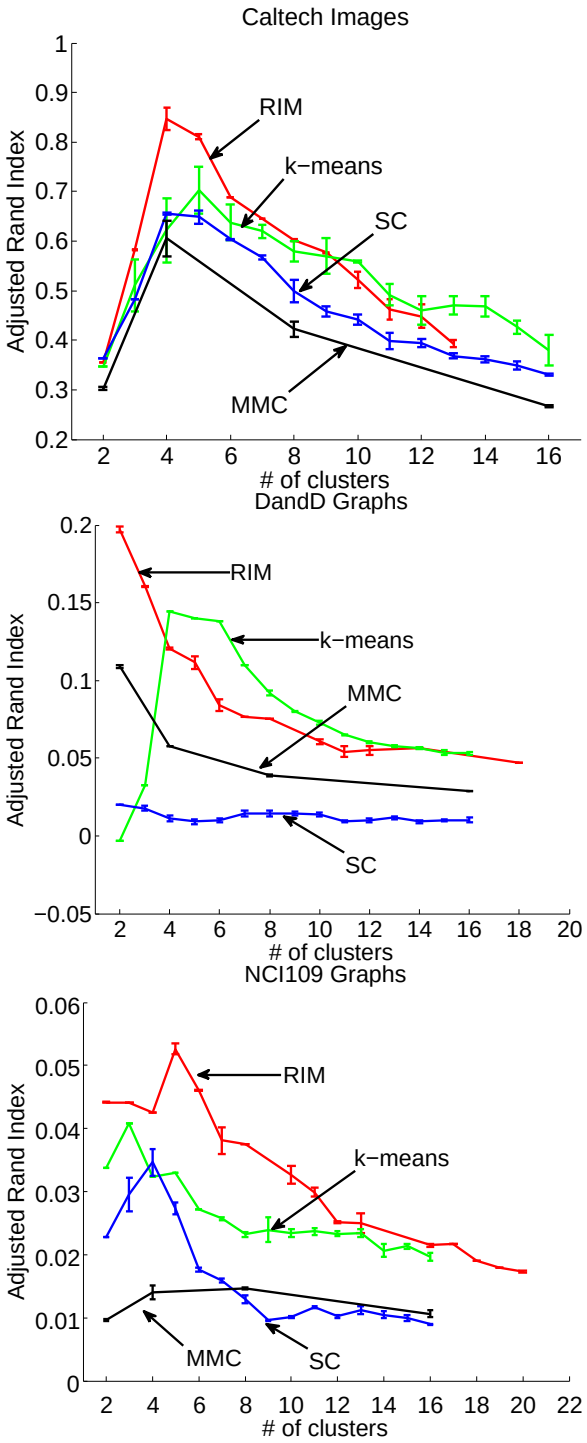


Figure 5.3: Unsupervised Clustering: Adjusted Rand Index measures the similarity of inferred clusters to a reference ground truth clustering, and is plotted as a function of the number of inferred clusters. Results shown for Caltech Images (top), D&D molecular graphs (center), and NCI109 (bottom) datasets. We find that our method (RIM) outperforms max-margin clustering (MMC) and spectral clustering (SC) at all settings. Our method outperforms k-means when discovering small numbers of clusters, and the performances tend to converge when the algorithms discover large numbers of clusters.

Image Clustering. We test the algorithms on an image clustering task with 350 images from four Caltech-256 [GHP07] categories (Faces-Easy, Motorbikes, Airplanes, T-Shirt) for a total of $N = 1400$ images. We use the Spatial Pyramid Match kernel [LSP06] computed between every pair of images. We sweep RIM's λ parameter across $[\frac{0.125}{N}, \frac{4}{N}]$. The results are summarized in Figure 5.3 (top). Overall, the clusterings that best match ground truth are given by RIM when it discovers four clusters. We find that RIM outperforms both SC and MMC at all settings. RIM outperforms kernelized k-means when discovering between 4 and 8 clusters. Their performances are comparable for other numbers of clusters. Figure 5.4 shows example images taken from clusters discovered by RIM. Our RIM implementation takes approximately 110 seconds per run on the Caltech Images dataset on a quad core Intel Xeon server. SC requires 38 seconds per run, while MMC requires 44–51 seconds per run depending on the number of clusters specified.

Molecular Graph Clustering. We further test RIM's unsupervised learning performance on two molecular graph datasets. D&D [DD03] contains $N = 1178$ protein structure graphs with binary ground truth labels indicating whether or not they function as enzymes. NCI109 [WK06] is composed of $N = 4127$ compounds labeled according to whether or not they are active in an anti-cancer screening. We use the subtree kernel developed by [SB10] with subtree height of 1. For D&D, we sweep RIM's lambda parameter through the range $[\frac{0.001}{N}, \frac{0.05}{N}]$ and for NCI we sweep through the interval $[\frac{0.001}{N}, \frac{1}{N}]$. Results are summarized in Figure 5.3 (center and bottom). We find that of all methods, RIM produces the clusterings that are nearest to ground truth (when discovering 2 clusters for D&D and 5 clusters for NCI109). RIM outperforms both SC and MMC at all settings. RIM has the advantage over k-means when discovering a small number of clusters and is comparable at other settings. On NCI109, RIM required approximately 10 minutes per run. SC required approximately 13 minutes, while MMC required on average 18 minutes per run.

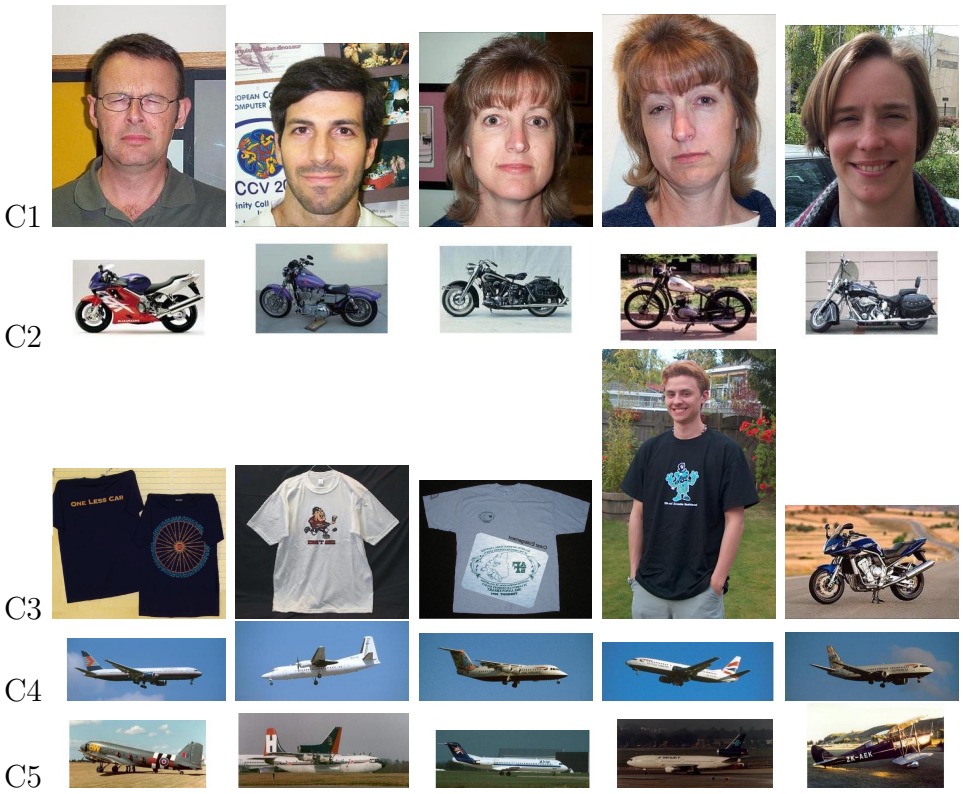


Figure 5.4: Randomly chosen example images from clusters discovered by unsupervised RIM on Caltech Image. Clusters are composed mainly of images from a single ground truth category. When RIM splits a ground truth category into two clusters (e.g. clusters C4 and C5), it does so along perceptually relevant lines. Here, C4 contains images of airplanes in the sky, and C5 contains images of airplanes on the ground.

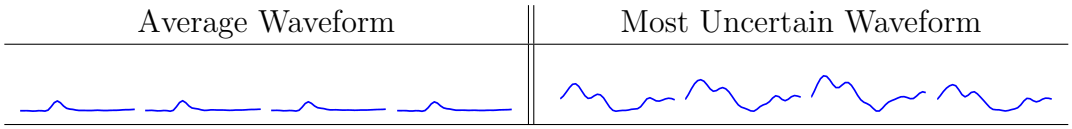


Figure 5.5: Left, Tetrode dataset average waveform. Right, the waveform with the most uncertain cluster membership according to the classifier learned by RIM

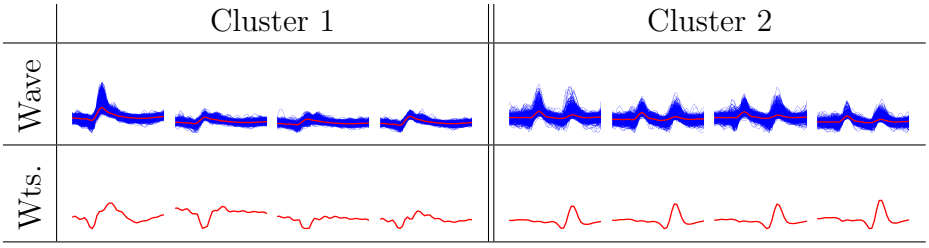


Figure 5.6: Two clusters discovered by RIM on the Tetrode data set. Top row: Superimposed waveform members, with cluster mean in red. Bottom row: The discriminative category weights $\mathbf{w}_k$ associated with each cluster. The discriminative weights indicate how the cluster's members differ from the average waveform.

Neural Tetrode Recordings. We demonstrate RIM on a large scale data set of $319,209$ neural activity waveforms recorded from four co-located electrodes implanted in the hippocampus of a behaving rat. The waveforms are composed of 38 samples from each of the four electrodes and are the output of a neural spike detector which aligns signal peaks to the 13th sample; see the average waveform in Figure 5.5 (left). We concatenate the samples into a single 152-dimensional vector and preprocess by subtracting the mean waveform and divide each vector component by its variance.

We use the linear RIM algorithm given in Section 5.4, initialized with 100 categories. We set λ to $\frac{4}{N}$ and RIM discovers 33 clusters and finishes in 12 minutes. There is no ground truth available for this dataset, but we use it to demonstrate RIM's efficacy as a data exploration tool. Figure 5.6 shows two clusters discovered by RIM. The top row consists of cluster member waveforms superimposed on each other, with the cluster's mean waveform plotted in red. We find that the clustered waveforms have substantial similarity to each other. Taken as a whole, the clusters give an idea of the typical waveform patterns. The bottom row shows the learned classifier's discriminative weights $\mathbf{w}_k$ for each category, which can be used to gain

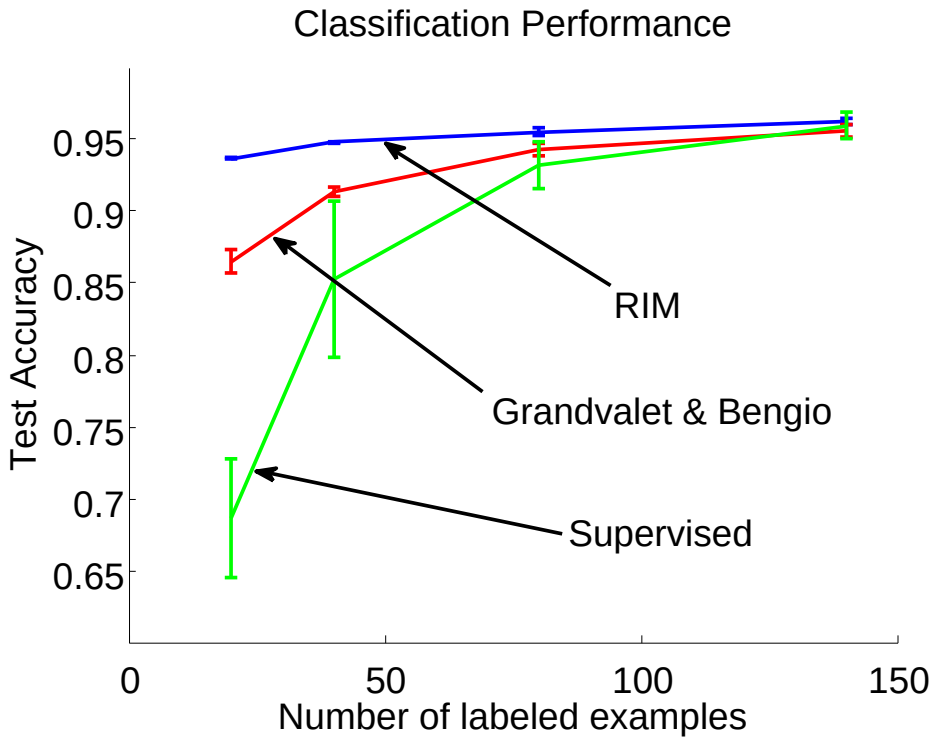


Figure 5.7: Semi-supervised learning on Caltech Images. Unlabeled examples improve test performance significantly. Our method makes use of prior information on class size proportions, which leads to a gain in performance relative to [GB04].

a sense for how the cluster's members differ from the dataset mean waveform. We can use the probabilistic classifier learned by RIM to discover atypical waveforms by ranking them according to their conditional entropy $H\{p(y|\mathbf{x}_i, \mathbf{W})\}$. Figure 5.5 (right) shows the waveform whose cluster membership is most uncertain.

5.7.2 Semi-Supervised Classification

We test our semi-supervised classification method described in Section 5.6.1 against [GB04] on the Caltech Images dataset. The methods were trained using both unlabeled and labeled examples, and classification performance is assessed on the unlabeled portion. As a baseline, a supervised classifier was trained on labeled subsets of the data and tested on the remainder. Parameters were selected via cross-validation on a subset of the labeled examples. The results are summarized in Figure 5.7. We find that both semi-supervised methods significantly improve classification performance relative to

the supervised baseline when the number of labeled examples is small. Additionally, we find that RIM outperforms Grandvalet & Bengio. This suggests that incorporating prior knowledge about class size distributions (in this case, we use a uniform prior) may be useful in semi-supervised learning.

5.8 Related Work

Our work has connections to existing work in both unsupervised learning and semi-supervised classification.

Unsupervised Learning. The information bottleneck method [TPB00] learns a conditional model $p(y|x)$ where the labels y form a lossy representation of the input space x, while preserving information about a third "relevance" variable z. The method maximizes $I(y;z) - \lambda I(x;y)$, whereas we maximize the information between y and x while constraining complexity with a parametric regularizer. The method of [SATB05] aims to maximize a similarity measure computed between members within the same cluster while penalizing the mutual information between the cluster label y and the input x. Again, mutual information is used to enforce a lossy representation of $y|x$. Song et al. [SSGB07] also view clustering as maximization of the dependence between the input variable and output label variable. They use the Hilbert-Schmidt Independence Criterion as a measure of dependence, whereas we use Mutual Information.

There is also an unsupervised variant of the Support Vector Machine, called max-margin clustering. Like our approach, the works of [XS05] and [BH07] use notions of class balance, seperation, and regularization to learn unsupervised discriminative classifiers. However, they are formulated in the max-margin framework rather than our probabilistic approach. Ours appears more amenable to incorporating prior beliefs about the class labels. Unsupervised SVMs are solutions to a convex relaxation of a non-convex problem, while we directly optimize our non-convex objective. The semidefinite programming methods required are much more expensive than our ap-

proach.

Semi-supervised Classification. Our semi-supervised objective is related to [GB04], as discussed in Section 5.6.1. Another semi-supervised method [CJ03] uses mutual information as a regularizing term to be minimized, in contrast to ours which attempts to maximize mutual information. The assumption underlying [CJ03] is that any information between the label variable and unlabeled examples is an artifact of the classifier and should be removed. Our method encodes the opposite assumption: there may be variability (e.g., new class label values) not captured by the labeled data, since it is incomplete.

5.9 Conclusions

We considered the problem of learning a probabilistic discriminative classifier from an unlabeled data set. We presented Regularized Information Maximization (RIM), a probabilistic framework for tackling this challenge. Our approach consists of optimizing an intuitive information theoretic objective function that incorporates class separation, class balance, and classifier complexity, which may be interpreted as maximizing the mutual information between the empirical input and implied label distributions. The approach is flexible, in that it allows consideration of different likelihood functions. It also naturally allows expression of prior assumptions about expected label proportions by means of a cross-entropy with respect to a reference distribution. Our framework allows natural incorporation of partial labels for semi-supervised learning. In particular, we instantiate the framework to unsupervised, multi-class kernelized logistic regression. Our empirical evaluation indicates that RIM outperforms existing methods on several real data sets, and demonstrates that RIM is an effective model selection method.

Part III

Postscript

Chapter 6

Closing Thoughts on Open Ended Learning

In this thesis, we have explored unsupervised learning as a foundation for autonomous categorization systems that can continuously learn and refine their collection of categories over time, while remaining computationally tractable. What are the prospects for practical object recognition systems capable of functioning in this way? As envisioned in this thesis, an Open Ended learning system makes use of a fixed data representation, upon which categories are defined. As might be expected, a system constructed in this way can only perform as well as its data representation allows. Specifically, categories must correspond to clusters of data items in the (implicit) data representation space.

Our experiments in unsupervised visual object categorization (see Section 5.4 and Section 2.5) made use of the Spatial Pyramid Match Kernel [LSP06] as the underlying data representation. This is a simple approach that encodes texture and an image's loose geometric properties. While we find that we can accurately recover simple categories with large numbers of training examples, extending these experiments to larger numbers of more complex visual categories [GHP07] proved impossible. Our hope in exploring discriminative clustering (see Chapter 5) was that rich category representations (defined by discriminative classifiers rather than parametric distributions) would lead to better extraction of categories. While our experiments bear this out to a certain extent, it is clear that the cluster assumption (see Section 5.3) is still

limiting as it often does not hold for visually complex categories in this representation. In other cases there were too few training examples to define regions of space with meaningful density.

This leads us to conclude that data representation (rather than category representation) is the key limitation in achieving Open Ended learning. Therefore, any research breakthrough in terms of representation of visual objects will lead to improved prospects for Open Ended learning. In fact, it is our opinion that issues of image representation are the key problem of computer vision today. In addition, progress might be made by removing the restriction of a fixed data representation all together. Unsupervised learning may be used to tune the lower level image computations in order to change the representation over time. While we have focused on learning from independently sampled data, it may be possible to learn an object representation without supervision by making use of temporally coherent video sequences, since there is evidence that the human visual system makes use of dynamic cues during learning [SOM06].

It is likely that, for the forseeable future, autonomous categorization systems will require significant human intervention in order to overcome limitations in the underlying data representation.[1] We propose two research directions focused on making the best use of limited human intervention in Open Ended learning systems:

- **Incremental Data Representation Learning.** Since we conclude that data representation is the key bottleneck towards Open Ended learning, we propose to use human provided categorical labels to adjust this representation itself. The representation should be adjusted so that data items that are labeled with the same category are 'close' to each other, while items that are in different categories are 'far' from each other. Essentially, we propose to use human labeling effort to improve the cluster assumption so that categories can then be defined as clusters of unlabeled points. See [LCB+04] and [XNJR02] for ex-

[1] Additionally, meaningful category distinctions are likely dependent on the overall functional task supported by the categorization system. External reinforcement or intervention may be necessary in defining functionally useful categories.

amples of kernel and metric learning, respectively. We require algorithms that incrementally update the data representation as labeled instances are acquired. A moment's thought reveals an additional computational difficulty: each time the data representation changes, the implied location of each unlabeled point also changes. We therefore require clever methods that can accomplish these updates and appropriately adjust categories without violating the computational resource budgets required in Open Ended learning.

- **Incremental Active Learning.** The category labels of some data items may be more informative than others, and ideally an Open Ended learning system would select the most informative examples for human labeling. We advocate research in *active learning* [TC01] specifically focused on the problems inherent in continuously adapting categorization systems. Much existing work in active learning takes a *pool based* approach [MN98], in which a representative pool of unlabeled examples is available all at once. In contrast, we require active learning algorithms compatible with incremental learning systems that do not have access to a fully representative pool. We also require algorithms tailored to the case where the number of categories is unknown which, to the best of our knowledge, has yet to be explored.

A complete system would ideally combine both research directions.

We conclude by suggesting that automated categorization systems should make use of interactions between humans and machines in order to improve performance. While this thesis has been concerned with minimizing the requirement for human intervention, the final chapter of this thesis explores a system that lies on the other end of the spectrum of human-machine interaction. In this work, humans are treated as fundamental computational building blocks and the machine takes the supervisory role of distributing tasks, as well as evaluating and aggregating the results of human workers.

Chapter 7

Crowd Clustering

7.1 Introduction

The Internet has enabled a new means of outsourcing information processing tasks to large groups of anonymous workers. *Crowdsourcing* services, such as Amazon's Mechanical Turk, have emerged as a convenient way to purchase *Human Intelligence Tasks* (HITs). Recently, the computer vision community has leveraged this capability to annotate large sets of images and video in order to use them as training data for supervised learning of computer vision systems [SF08].

Here, we instead examine the possibility of using crowdsourcing services to perform large scale data analysis tasks which are typically the domain of automated systems. Our question is: can we use crowdsourcing services to reliably categorize large collections of human-interpretable patterns? We therefore view crowdsourcing as a means of carrying out distributed human computation [Zit08]. Our goal is to distribute a categorization task among a large set of workers, and then to use an automated algorithm to assemble the individual HITs into a complete solution. This may be viewed as an inversion of the typical machine learning paradigm in which computers do the majority of the work, overseen by a human supervisor. Here, humans are treated as the basic computational building block and a machine acts as a supervisor that distributes tasks and evaluates the results.

7.2 The Crowd Clustering Problem

Suppose we possess a large collection of unorganized images and wish to categorize them into clusters of related images. How can we hope to accomplish this? Since we have a large number (perhaps several thousand or more) of images, it is not realistic to expect a single person to perform the categorization task. We may choose to use an automated data clustering algorithm to perform the task, yet unsupervised categorization of images is unfortunately a problem that is far from solved. We can not reasonably expect an automated algorithm to organize the images in a satisfactory fashion. Here, we explore the possibility of dividing the images into groups of reasonable size, and distributing them as HITs to a large pool of human workers. The workers will then perform the HITs by categorizing these images into clusters. Finally, we will aggregate these restricted clusterings into a complete clustering of the full dataset.

We propose *Crowd Clustering*, an approach to clustering that may be used when the set of data items have the following characteristics:

- The data items are human-interpretable such that people are able to group them into sensible categories, yet no automated clustering algorithm is known to perform well on them. The set of data may be images of complex scenes or objects, passages of text, audio samples, or other type of pattern. Unfortunately, arbitrary high dimensional data vectors are likely excluded since humans may be unable to naturally group them.

- The set of data is too large for a single human to perform the categorization task. This is likely to be the case when the items number in the thousands or more.

- Human workers may have different ideas or schools of thought about how to organize the items. For example, workers may choose to group objects according to different attributes such as height or color. Workers may also have different notions about the number of categories. Given a collection of natural scenes,

one worker might decide to group them according to whether they are taken indoors or outdoors. Another may choose to draw finer distinctions, dividing indoor images into kitchen, living room, and bedroom categories.

Three questions naturally arise. The first is: How do we best structure the tasks that we assign to workers, and what type of information do we expect workers to provide? Existing work in data visualization, known as Multidimensional Scaling (MDS) [Kru64] makes use of human provided similarity values defined between pairs of data items. In practice, these distances may be elicited from workers by asking them to rate the similarity of a pair of objects on a discrete scale. Because we are explicitly interested in forming categories, we choose instead to structure HITs as a categorization task of M items, presented simultaneously. The worker then groups the M items into clusters of his choosing. In general, we do not explicitly predefine categories, the user is free to cluster the items into as many or few groups as he sees fit. Likewise, an item may be placed in its own cluster if it is unlike the others in the HIT.

We do not expect different workers to agree on their definitions of categories. In fact, even a single worker may not be entirely consistent in his notion of categories when performing multiple HITs. We therefore avoid the complex problem of explicitly associating categories across HITs. Instead, we represent the results of a HIT as a series of $M(M-1)/2$ binary labels defined between all pairs of images in the HIT. If two items are grouped in the same cluster in the HIT, then their associated pairwise label takes the value 1, if they are in different groups then the label takes value -1.

The next question is: How do we best divide the data items into reasonable size collections when distributing them to workers? Our problem may be considered an *object distributed* clustering problem [SG02], where we must work with clusterings of only a subset of the data. [SG02] use a random sampling scheme to divide data items into groups before clustering them with automated algorithms. Their scheme controls the level of sampling redundancy (the degree to which a data item is present in multiple clustering tasks) with a single parameter. With the exception of this method, we find a relative dearth of ideas in the machine learning literature about

tackling this problem. We also expect that ideas from human psychology may inform our approach.

Finally: Once we have obtained clusterings from workers, how do we aggregate them into a single clustering? There is an extensive literature in machine learning on the problem of combining multiple alternative clusterings of data. This problem is known as consensus clustering [MTMG03], clustering aggregation [GMT07], or cluster ensembles [SG02]. However, existing approaches focus on producing a single "average" clustering from a set of clusterings.

In contrast, we are not merely interested in the average clustering produced by a crowd of workers. Instead, we are interested in understanding the ways in which different individuals may categorize the data. We wish to produce a master clustering of the data that may be combined in different ways in order to describe the tendencies of individual workers. We refer to these groups of data as *platonic* clusters, since they are meta-concepts that may be used to form the workers' differing notions of categories.

For example, suppose one worker groups objects into a cluster of tall objects and another of short objects, while a different worker groups the same objects into a cluster of red objects and another of blue objects. Then, our method should recover four platonic clusters: tall red objects, short red objects, tall blue objects, and short blue objects. The behavior of the two workers may then be summarized using a confusion table of the platonic clusters (see Section 7.3.1.2). The first worker groups the first and third platonic cluster into one category and the second and fourth platonic cluster into another category. The second worker groups the first and second platonic clusters into a category and the third and fourth platonic clusters into another category.

7.2.1 Notation

Before continuing to our technical approach, we define some relevant notation involving vectors, matrices, and tensors. $\mathbf{1}$ represents the matrix with value 1 in every entry, and $\mathbf{I}$ represents the identity matrix. The notation $[\mathbf{v}]_d$ refers to the d-th entry

of vector $\mathbf{v}$ and $[\mathbf{M}]_{d_1 d_2}$ is the entry at row d_1 and column d_2 in matrix $\mathbf{M}$. We extend this notation to higher order tensors (e.g., $[\mathbf{T}]_{d_1 d_2 d_3 d_4}$ refers to the element at location (d_1, d_2, d_3, d_4) of fourth order tensor $\mathbf{T}$.) The operator $\mathbf{v} = \text{diag}\{\mathbf{M}\}$ yields a column vector with the diagonal elements of $\mathbf{M}$, that is $[\mathbf{v}]_d = [\mathbf{M}]_{dd}$. $\mathbf{A} \circ \mathbf{B}$ refers to element-wise multiplication of vectors, matrices, or tensors. The operator $\mathbf{v} = \text{vecp}\{\mathbf{M}\}$ yields a column vector $\mathbf{v}$ from the upper triangular portion of $\mathbf{M}$ by "stacking" the partial columns of $\mathbf{M}$, according to the ordering $[\mathbf{v}]_1 = [\mathbf{M}]_{11}, [\mathbf{v}]_2 = [\mathbf{M}]_{12}, [\mathbf{v}]_3 = [\mathbf{M}]_{22}$, etc.

7.3 Our Approach

We give two methods for aggregating the results of HITs as well as modeling the specific characteristics of each worker. We assume that there are N total items (indexed by i) and J total workers (indexed by j.) These methods take as input the collection of binary labels produced by each HIT. Formally, we treat this as a set of binary variables $\mathcal{L}$, with elements $l_t \in \{-1, +1\}$ indexed by a positive integer t. Associated with the t-th label is a triple (a_t, b_t, j_t), where $j_t \in \{1, \ldots, J\}$ indicates the worker that produced the label, and $a_t \in \{1, \ldots, N\}$ and $b_t \in \{1, \ldots, N\}$ indicate the two data items compared by the label.

7.3.1 Bayesian Crowd Clustering

Here we propose an approach in which data items are represented as points in a Euclidean space and workers are modeled as binary classifiers in this space. Platonic clusters are then obtained by clustering these inferred points using the mixture model approach from Ch. 2. The advantage of an intermediate Euclidean representation is that it provides a compact way to capture the characteristics of each data item. For example, two images may be similar along certain axes (perhaps relating to an object's color) but different along another (e.g., the object's pose.) A similar approach was proposed by Welinder et al. [WBBP10] for the analysis of classification labels obtained from crowdsourcing services. This method does not apply to our problem, since it involves binary labels applied to single data items rather than to pairs, as in our case.

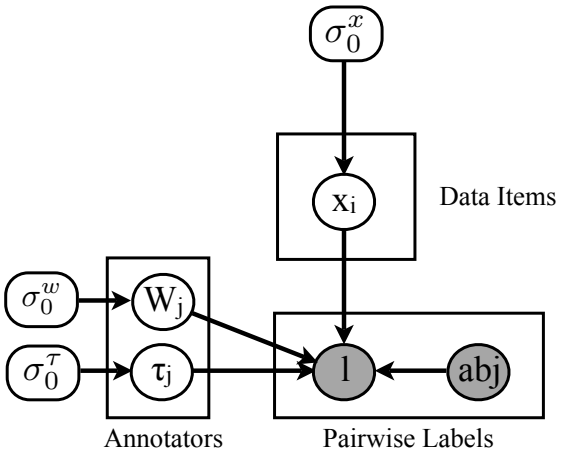


Figure 7.1: Our model for modeling image and worker characteristics in the Crowd Clustering problem. Known variables are colored gray, and fixed hyper-parameters are given in rounded boxes.

Their method therefore requires that categories be defined a priori and agreed upon by all workers, which is incompatible with the Crowd Clustering problem.

We propose a novel probabilistic latent variable model that relates pairwise binary labels to hidden variables associated with both workers and images. The graphical model is shown in Figure 7.1. $\mathbf{x}_i$ is a D dimensional vector, which encodes item i's location in the platonic space $\mathbb{R}^D$. Symmetric matrix $\mathbf{W}_j \in \mathbb{S}^{\mathbb{D}\times\mathbb{D}}$ and bias $\tau_j \in \mathbb{R}$ are used to model worker j's behavior.

The joint distribution is

$$p(X, W, \tau, \mathcal{L}) = \prod_i p(\mathbf{x}_i|\sigma_0^x) \prod_j p(\mathrm{vecp}\{\mathbf{W}_j\}|\sigma_0^w) p(\tau_j|\sigma_0^\tau) \prod_t p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}). \tag{7.3.1}$$

The conditional distributions are defined as follows:

$$p(\mathbf{x}_i|\sigma_0^x) = \prod_d \mathrm{Normal}([\mathbf{x}_i]_d; 0, \sigma_0^x) \tag{7.3.2}$$

$$p(\mathrm{vecp}\{\mathbf{W}_j\}|\sigma_0^w) = \prod_{d_1 \le d_2} \mathrm{Normal}([\mathbf{W}_j]_{d_1 d_2}; 0, \sigma_0^w)$$

$$p(\tau_j|\sigma_0^\tau) = \text{Normal}(\tau_j; 0, \sigma_0^\tau),$$
$$p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}) = \frac{1}{1+\exp(-l_t A_t)}$$

where $(\sigma_0^x, \sigma_0^\tau, \sigma_0^w)$ are fixed hyper-parameters, and we define the *activity*:

$$A_t = \mathbf{x}_{a_t}^T \mathbf{W}_{j_t} \mathbf{x}_{b_t} + \tau_{j_t}. \tag{7.3.3}$$

The key term is the pairwise quadratic logistic regression likelihood (which is novel to the best of our knowledge) that captures worker j's tendency to label the pair of images a_t and b_t with l_t. Symmetry of $\mathbf{W}_j$ ensures that $p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}) = p(l_t|\mathbf{x}_{b_t}, \mathbf{x}_{a_t}, \mathbf{W}_{j_t}, \tau_{j_t})$. This form of likelihood yields a compact and tractable method of representing classifiers defined over pairs of points in Euclidean space. In practice, we find that our algorithm (see Section 7.3.1.1) tends to find positive definite matrices $\mathbf{W}_j$ associated with each worker, which define an inner product $\mathbf{x}_a^T \mathbf{W}_j \mathbf{x}_b$ between a pair of vectors. Pairs of vectors with large inner product tend to be classified as being in the same category, and in different categories otherwise. (We choose not to explicitly enforce a positive definite constraint on $\mathbf{W}_j$, since it would significantly increase algorithmic complexity to do so and we find it unnecessary for our purpose.)

We find that this form of likelihood leads to tightly grouped clusters of points $\mathbf{x}_i$ that are then easily discovered by mixture model clustering. In principle, mixture of gaussian clusters could be estimated jointly by extending our graphical model to include them (Figure 7.2), and we will explore this in the future.

7.3.1.1 Algorithm

Exact posterior inference in this model is known to be intractable, since computing it involves integrating over variables with complex dependencies. We therefore develop an inference algorithm based on the Variational Bayes method [Att99]. The high level idea is to work with a factorized proxy posterior distribution that does not model the full complexity of interactions between variables; it instead represents a single mode of the true posterior. Because this distribution is factorized, integrations involving it

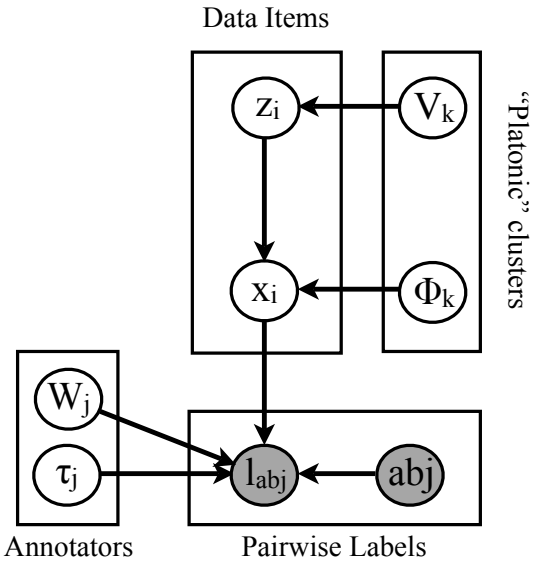


Figure 7.2: Extended Crowd Clustering model with joint estimation of platonic clusters. Hyper-parameters omitted for clarity.

become tractable. We define the proxy distribution

$$q(X, W, \tau) = \prod_i q(\mathbf{x}_i; \boldsymbol{\mu}_i^x, \boldsymbol{\sigma}_i^x) \prod_j q(\text{vecp}\{\mathbf{W}_j\}; \boldsymbol{\mu}_j^w, \boldsymbol{\sigma}_j^w) q(\tau_j; \mu_j^\tau, \sigma_j^\tau) \tag{7.3.4}$$

using parametric distributions of the following form:

$$\begin{aligned} q(\mathbf{x}_i; \boldsymbol{\mu}_i^x, \boldsymbol{\sigma}_i^x) &= \prod_d \text{Normal}([\mathbf{x}_i]_d; [\boldsymbol{\mu}_i^x]_d, [\boldsymbol{\sigma}_i^x]_d) \\ q(\text{vecp}\{\mathbf{W}_j\}; \boldsymbol{\mu}_j^w, \boldsymbol{\sigma}_j^w) &= \prod_{d_1 \le d_2} \text{Normal}([\mathbf{W}_j]_{d_1 d_2}; [\boldsymbol{\mu}_j^w]_{d_1 d_2}, [\boldsymbol{\sigma}_j^w]_{d_1 d_2}) \\ q(\tau_j; \mu_j^\tau, \sigma_j^\tau) &= \text{Normal}(\tau_j; \mu_j^\tau, \sigma_j^\tau). \end{aligned} \tag{7.3.5}$$

$\boldsymbol{\mu}_i^x$ and $\boldsymbol{\sigma}_i^x$ are variational mean and variance parameters associated with data item i. $\boldsymbol{\mu}_j^w$ and $\boldsymbol{\sigma}_j^w$ are symmetric matrix variational mean and variance parameters associated with worker j, and μ_j^τ and σ_j^τ are variational mean and variance parameters for the bias τ_j of worker j. We use diagonal covariance Normal distributions over $\mathbf{W}_j$ and $\mathbf{x}_i$ in order to reduce the number of parameters that must be estimated.

Next, we define a cost function which allows us to determine the variational pa-

rameters. We use Jensen's inequality in order to develop a lower bound to the log evidence:

$$\begin{aligned}
&\log p(\mathcal{L}|\sigma_0^x, \sigma_0^\tau, \sigma_0^w) \qquad (7.3.6)\\
&= \log \int p(X, W, \tau, \mathcal{L}) dX dW d\tau \\
&= \log \int q(X, W, \tau) \frac{p(X, W, \tau, \mathcal{L})}{q(X, W, \tau)} dX dW d\tau \\
&\geq \int q(X, W, \tau) \log \frac{p(X, W, \tau, \mathcal{L})}{q(X, W, \tau)} dX dW d\tau \\
&= E_q \log p(X, W, \tau, \mathcal{L}) + \mathcal{H}\{q(X, W, \tau)\}.
\end{aligned}$$

$\mathcal{H}\{q(X, W, \tau)\}$ is the entropy of the proxy distribution, and $E_q \log p(X, W, \tau, \mathcal{L}) + \mathcal{H}\{q(X, W, \tau)\}$ is know as the *Free Energy*. It can be shown that the difference between the log evidence and the Free Energy lower bound is

$$KL\{q(X, W, \tau)||p(X, W, \tau|L, \sigma_0^x, \sigma_0^\tau, \sigma_0^w)\}.$$

Therefore, maximizing the lower bound corresponds to minimizing the KL divergence between the proxy distribution and the true posterior.

However, the Free Energy above still involves intractable integration, because the normal distribution priors over variables $\mathbf{W}_j$, $\mathbf{x}_i$, and τ_j are not conjugate [BS94] to the logistic likelihood term. We therefore substitute the left hand side of the following inequality for the likelihood

$$g(\Delta_t) \exp\{(l_t A_t - \Delta_t)/2 + \lambda(\Delta_t)(A_t^2 - \Delta_t^2)\} \leq p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}) \qquad (7.3.7)$$

which is adapted from [JJ96] to our case of quadratic pairwise logistic regression, in order to obtain a fully tractable lower bound. Here $g(x) = (1 + e^{-x})^{-1}$ and $\lambda(\Delta) = [1/2 - g(\Delta)]/(2\Delta)$. This expression introduces an additional variational parameter Δ_t for each label, which are optimized in order to tighten the lower bound.

Our cost function is therefore:

$$\begin{aligned}\mathcal{F} =& E_q \log p(X, W, \tau) + \mathcal{H}\{q(X, W, \tau)\} \\ &+ \sum_t \log g(\Delta_t) + \frac{l_t}{2} E_q\{A_t\} - \frac{\Delta_t}{2} + \lambda(\Delta_t)(E_q\{A_t^2\} - \Delta_t^2)\end{aligned} \tag{7.3.8}$$

Intermediate terms required to evaluate this expression are given in this chapter's Appendix (Eq. 7.7.4).

Optimization of variational parameters is carred out in a coordinate ascent procedure, which exactly maximizes each variational parameter in turn while holding all others fixed. It is guaranteed to converge to a local maximum of the cost function. The update equations are

$$\begin{aligned}[\boldsymbol{\sigma}_i^x]_d &= \Big(1/\sigma_0^x + \sum_{t:a_t=i} 2|\lambda(\Delta_t)|\big[E_q\{\mathbf{W}_{j_t}\mathbf{x}_{b_t}\mathbf{x}_{b_t}^T\mathbf{W}_{j_t}\}\big]_{dd} \\ &\qquad + \sum_{t:b_t=i} 2|\lambda(\Delta_t)|\big[E_q\{\mathbf{W}_{j_t}\mathbf{x}_{a_t}\mathbf{x}_{a_t}^T\mathbf{W}_{j_t}\}\big]_{dd}\Big)^{-1} \\ \boldsymbol{\mu}_i^x &= (\mathbf{I} - \mathbf{U}_i \circ (\mathbf{1} - \mathbf{I}))^{-1}\mathbf{v}_i \\ \sigma_j^\tau &= \Big(1/\sigma_0^\tau + \sum_{t:j_t=j} 2|\lambda(\Delta_t)|\Big)^{-1} \\ \mu_j^\tau &= \sigma_j^\tau \sum_{t:j_t=j} l_t/2 + 2\lambda(\Delta_t)(\boldsymbol{\mu}_{a_t}^x)^T\boldsymbol{\mu}_j^w\boldsymbol{\mu}_{b_t}^x \\ [\boldsymbol{\sigma}_j^w]_{d_1d_2} &= \left(1/\sigma_0^w + \sum_{t:j_t=j} 2|\lambda(\Delta_t)|\big[E_q\{\mathbf{Y}^{a_tb_t}\}\big]_{d_1d_2d_1d_2}\right)^{-1} \\ \mathrm{vecp}\{\boldsymbol{\mu}_j^w\} &= (\mathbf{I} - \mathbf{B}_j \circ (\mathbf{1} - \mathbf{I}))^{-1}\mathbf{c}_j \\ \Delta_t &= (E_q\{A_t^2\})^{1/2}\end{aligned} \tag{7.3.9}$$

Expressions for the intermediate quantities $\mathbf{B}_j$, $\mathbf{c}_j$, $\mathbf{U}_i$, $\mathbf{v}_i$, as well as the expecations $E_q\{\mathbf{W}_j\mathbf{x}_a\mathbf{x}_a^T\mathbf{W}_j\}$ and $E_q\{\mathbf{Y}^{ab}\}$ are given in the Appendix. We iterate the above update equations until Eq. 7.3.8 converges.

7.3.1.2 Worker Confusion Analysis

Upon convergence, we use a Dirichlet process mixture model to cluster the mean locations of the image items $\boldsymbol{\mu}_i^x$, and to determine the appropriate number of clusters. We use the mean parameter values $\Phi_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ of the discovered platonic clusters. We are interested in the predicted confusion matrix $\mathbf{C}_j$ for worker j, where

$$[\mathbf{C}_j]_{k_1 k_2} = \int p(l=1|\mathbf{x}_a, \mathbf{x}_b, \mathbf{W}_j, \tau_j) p(\mathbf{x}_a|\boldsymbol{\mu}_{k_1}, \boldsymbol{\Sigma}_{k_1}) p(\mathbf{x}_b|\boldsymbol{\mu}_{k_2}, \boldsymbol{\Sigma}_{k_2}) \\ q(\text{vecp}\{\mathbf{W}_j\}; \boldsymbol{\mu}_j^w, \boldsymbol{\sigma}_j^w) q(\tau_j; \mu_j^\tau, \sigma_j^\tau) d\mathbf{x}_a d\mathbf{x}_b d\mathbf{W}_j d\tau_j \tag{7.3.10}$$

which expresses the probability that worker j assigns data items sampled from platonic cluster k_1 and k_2 to the same cluster. $p(\mathbf{x}_a|\boldsymbol{\mu}_{k_1}, \Sigma_{k_1})$ and $p(\mathbf{x}_b|\boldsymbol{\mu}_{k_2}, \Sigma_{k_2})$ are Normal distributions. This integration is intractable, however, we can again use Jensen's inequality and Eq. 7.3.7 to yield a tractable lower bound. Maximizing this bound over Δ yields

$$[\hat{\mathbf{C}}_j]_{k_1 k_2} = g(\hat{\Delta}_{k_1 k_2 j}) \exp\{(\boldsymbol{\mu}_{k_1}^T \boldsymbol{\mu}_j^w \boldsymbol{\mu}_{k_2} + \mu_j^\tau - \hat{\Delta}_{k_1 k_2 j})/2\} \tag{7.3.11}$$

which we use as our approximate confusion matrix, where $\hat{\Delta}_{k_1 k_2 j}$ is given in the Appendix.

7.3.2 Crowd Clustering via Matrix Factorization

We develop an alternative approach to Crowd Clustering based on non-negative matrix factorization in order to form a baseline measure to compare against our Bayesian model. This approach does not learn a location in a Euclidean space for data items, but instead directly estimates their platonic cluster membership. In addition, here we explicitly specify the number of platonic clusters K. No other hyper-parameters are required.

We represent the binary labels l_t and their associated triples (a_t, b_t, j_t) in matrix

form, where there is an $N \times N$ matrix $\mathbf{L}_j$ associated with each worker. The entries of the matrix are the average binary label provided by worker j for the pair of data items (a, b):

$$[\mathbf{L}_j]_{ab} = \frac{1}{N_{jab}} \sum_{t:j_t=j \wedge [(a_t,b_t)=(a,b) \vee (a_t,b_t)=(b,a)]} (l_t + 1)/2, \tag{7.3.12}$$

where $N_{jab} = \#\{t : j_t = j \wedge [(a_t, b_t) = (a, b) \vee (a_t, b_t) = (b, a)]\}$ is the number of times that worker j has labeled the pair (a, b). Note that we map the binary labels from the set $\{-1, 1\}$ to $\{0, 1\}$, and also that matrices $\mathbf{L}_j$ are symmetric. We allow the label matrices $\mathbf{L}_j$ to be incomplete, so that worker j need not compare every pair of images. We define binary matrices $\mathbf{P}_j$, where entry $[\mathbf{P}_j]_{ab} = 1$ if worker j has labeled pair (a, b) at least once and $[\mathbf{P}_j]_{ab} = 0$, otherwise. Incomplete entries in $\mathbf{L}_j$ may take an arbitrary value.

It is useful to review the consensus clustering problem posed in [LDJ07]. Consensus clustering finds a clustering of the N data items that is close to the average of a number of alternative clusterings. Given a set of complete label matrices $\mathbf{L}_j$ (with no missing values), consensus clustering may be formulated as the following matrix factorization problem:

$$\begin{aligned} \underset{\mathbf{W},\mathbf{Z}}{\operatorname{argmin}} & \left\| \frac{1}{J} \sum_j \mathbf{L}_j - \mathbf{Z}\mathbf{W}\mathbf{Z}^T \right\|_{\mathcal{F}}^2 \\ \text{s.t. } & \mathbf{Z}^T\mathbf{Z} = \mathbf{I} \\ & \mathbf{W} \geq 0 \\ & \mathbf{Z} \geq 0. \end{aligned} \tag{7.3.13}$$

The $N \times K$ matrix $\mathbf{Z}$ is constrained to be orthogonal with non-negative entries, and [LDJ07] show that it may therefore be interpreted as a clustering, where $[\mathbf{Z}]_{ik} > 0$ means that data item i is a member of cluster k. The matrix $\mathbf{W}$ is a $K \times K$ diagonal matrix (although the diagonality constraint is relaxed in practice) where entry $[\mathbf{W}]_{kk}$ yields the number of data items in cluster k. The objective is to find a clustering

that is close (in the sense of Frobenius norm) to the average pairwise relations of the constituent clusterings: $\frac{1}{J}\sum_j \mathbf{L}_j$.

In Crowd Clustering, we are instead interested in learning a set of K platonic clusters that may be recombined to model the diverse categorization behavior of the workers. Each worker is modeled with a $K \times K$ confusion matrix $\mathbf{W}_j$ (no longer expected to be diagonal) and $\mathbf{Z}$ represents the platonic clustering. Assuming complete matrices $\mathbf{L}_j$, we formulate Crowd Clustering as the following optimization problem:

$$\begin{aligned} \underset{\forall j \mathbf{W}_j, \mathbf{Z}}{\operatorname{argmin}} \frac{1}{J}\sum_j \|\mathbf{L}_j - \mathbf{Z}\mathbf{W}_j\mathbf{Z}^T\|_{\mathcal{F}}^2 \qquad (7.3.14) \\ \text{s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I} \\ \forall j, \mathbf{W}_j \geq 0 \\ \mathbf{Z} \geq 0. \end{aligned}$$

While similar in form to the consensus clustering problem, Crowd Clustering finds a platonic clustering that is not merely close to the average behavior of the crowd, but one that may be used to facilitate modeling of the variety of behaviors exhibited by the workers.

7.3.2.1 Algorithm

Given a set of complete label matrices $\mathbf{L}_j$, iterative update equations for the optimization problem specified by Eqs. 7.3.14 may be developed by modifying the arguments given in [DLPP06]. However, we wish to operate using incomplete label matrices, so that we need not exhaustively sample every pair of images for each worker. We take the EM-like approach of [ZWFM06] developed for non-negative matrix factorization of incomplete matrices. We first complete the label matrices using the current estimates of $\mathbf{Z}$ and $\mathbf{W}_j$ (akin to the 'E-step' of the EM algorithm):

$$\hat{\mathbf{L}}_j = \mathbf{P}_j \circ \mathbf{L}_j + (\mathbf{1} - \mathbf{P}_j) \circ \mathbf{Z}\mathbf{W}_j\mathbf{Z}^T. \qquad (7.3.15)$$

We then update $\mathbf{Z}$ and $\mathbf{W}_j$ using the completed label matrices $\hat{\mathbf{L}}_j$ (akin to the 'M-step' of the EM algorithm):

$$[\mathbf{Z}]_{ik} = [\mathbf{Z}]_{ik} \sqrt{\frac{\left[\sum_j \hat{\mathbf{L}}_j \mathbf{Z} \mathbf{W}_j\right]_{ik}}{\left[\mathbf{Z}\mathbf{Z}^T (\sum_j \hat{\mathbf{L}}_j \mathbf{Z} \mathbf{W}_j)\right]_{ik}}} \tag{7.3.16}$$

$$[\mathbf{W}_j]_{k_1 k_2} = [\mathbf{W}_j]_{k_1 k_2} \sqrt{\frac{\left[\mathbf{Z}^T \hat{\mathbf{L}}_j \mathbf{Z}\right]_{k_1 k_2}}{\left[\mathbf{Z}^T \mathbf{Z} \mathbf{W}_j \mathbf{Z}^T \mathbf{Z}\right]_{k_1 k_2}}}.$$

We alternate Eq. 7.3.15 and Eqs. 7.3.16 until convergence.

7.4 Sampling Methods

We have given two methods for aggregating the categorization results of workers into a platonic clustering, as well as a means of analyzing worker behavior in terms of confusion matrices. However, as mentioned in Section 7.2, we must first decide how to distribute images to workers before we may aggregrate their results. Since we have chosen to structure HITs as clustering tasks of M data items, we must specify the M data items in each HIT.

As a baseline method, we use the distributed sampling scheme outlined in [SG02] in order to distribute data items into HITs. This scheme is parameterized by a constant V, which designates the (expected) number of HITs in which a data item appears. In this approach, there are a total of $\lceil \frac{NV}{M} \rceil$ HITs. First, the N images are distributed deterministically among the HITs, so that there are $\lceil \frac{M}{V} \rceil$ items in each HIT. Then the remaining $M - \lceil \frac{M}{V} \rceil$ items in each HIT are filled by sampling without replacement from the $N - \lceil \frac{M}{V} \rceil$ items that are not yet allocated to the HIT.

We are currently exploring adaptive methods which attempt to gain more information with fewer HITs by selecting items intelligently. These methods have in common that they operate in a sequential fashion in which HITs in later rounds are constructed depending on the results from earlier rounds.

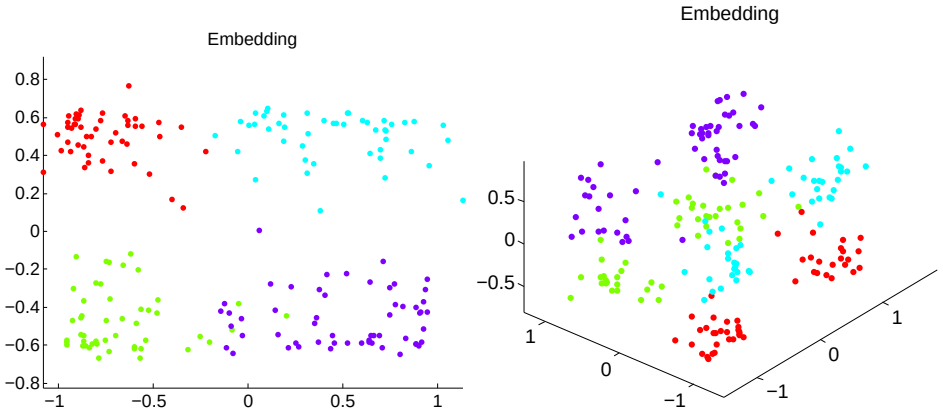


Figure 7.3: Embeddings $\boldsymbol{\mu}_i^x$ learned via Bayesian Crowd Clustering for the Greeble dataset. The color of the datapoint indicates its ground truth cluster label. In two dimensions (left), the four ground truth clusters are reproduced. In three dimensions (right) the original four clusters are split into eight clusters along an additional axis, capturing the uninstructed tendency of some workers to categorize based on Greeble dot color.

7.5 Experiments

In our preliminary experiments we generate HITs using the sampling scheme outlined in Section 7.4, using the value $V = 6$. We then distribute these HITs to Amazon Mechanical Turk such that each HIT is completed by 10 different workers.

7.5.1 Greebles

As a synthetic baseline experiment, we generate $N = 200$ "Greeble" images (see examples in Figure 7.4) using a method which maps a two-dimensional vector (g_1, g_2) to an image. Coordinate $g_1 \in [-1, 1]$ controls the Greeble's height (with 1 indicating tall and -1 indicating short) and $g_2 \in [-1, 1]$ controls the color of the Greeble's body (with 1 indicating a green body and -1 indicating a yellow body.) Dots, whose color (either blue or red) and size are randomly selected, are superimposed on the Greebles. We generate four clusters of Greebles from Gaussian distributions with means $(\mu_{g_1}, \mu_{g_2}) = (\pm 1, \pm 1)$ and standard deviation 0.1. Half of the workers were instructed to cluster Greebles based on their height, while the other half were instructed to cluster based on body color.

The resulting mean embedding $\boldsymbol{\mu}_i^x$ of the data items inferred by Bayesian Crowd Clustering are shown in Figure 7.3. In two dimensions ($D = 2$), we find that the four ground truth clusters are relatively well seperated, and the coordinate axes correspond to height and body color. In three dimensions ($D = 3$), we find that the clusters are split additionally along a third axis to yield a total of eight platonic clusters. No additional platonic clusters were discovered when $D > 3$.

Figure 7.4 shows high confidence exemplars from the nine platonic clusters discovered via mixture model clustering. We find that the additional axis of variability encodes the color of the Greeble's dots, which we did not instruct the workers to use as a categorization criterion. We observed similar platonic clusters when using the NMF Crowd Clustering method with $K = 8$. (Detailed comparison of the methods is left for future work.)

How do we explain this spontaneous "discovery" of the algorithm? Figure 7.5 shows the worker confusion matrices $\hat{\mathbf{C}}_j$ for three workers. As expected, many of the workers are shown to be sensitive to Greeble height and body color (which can be inferred based on examples from the platonic clusters that they assign to the same cluster.) However, there is a third type of worker that spontaneously chose to categorize based on dot color, despite the fact that we did not explicitly instruct them to do so. We find this result encouraging in that we are able to use our aggregation methods to interpret the variety of behaviors of different workers in the crowd.

7.5.2 Bird Pose

Our next experiment involves $N = 976$ images of Ring-billed Gulls from the Caltech-UCSD Birds-200 dataset [WBM$^+$10]. Workers were instructed to categorize the images according to pose, defined loosely as the viewpoint orientation or activity of the bird. Beyond that, no instructions were given in terms of the definition or number of pose categories.

Figures 7.6 and 7.7 show high confidence examples from platonic clusters inferred by Bayesian Crowd Clustering. The number of categories was inferred automatically

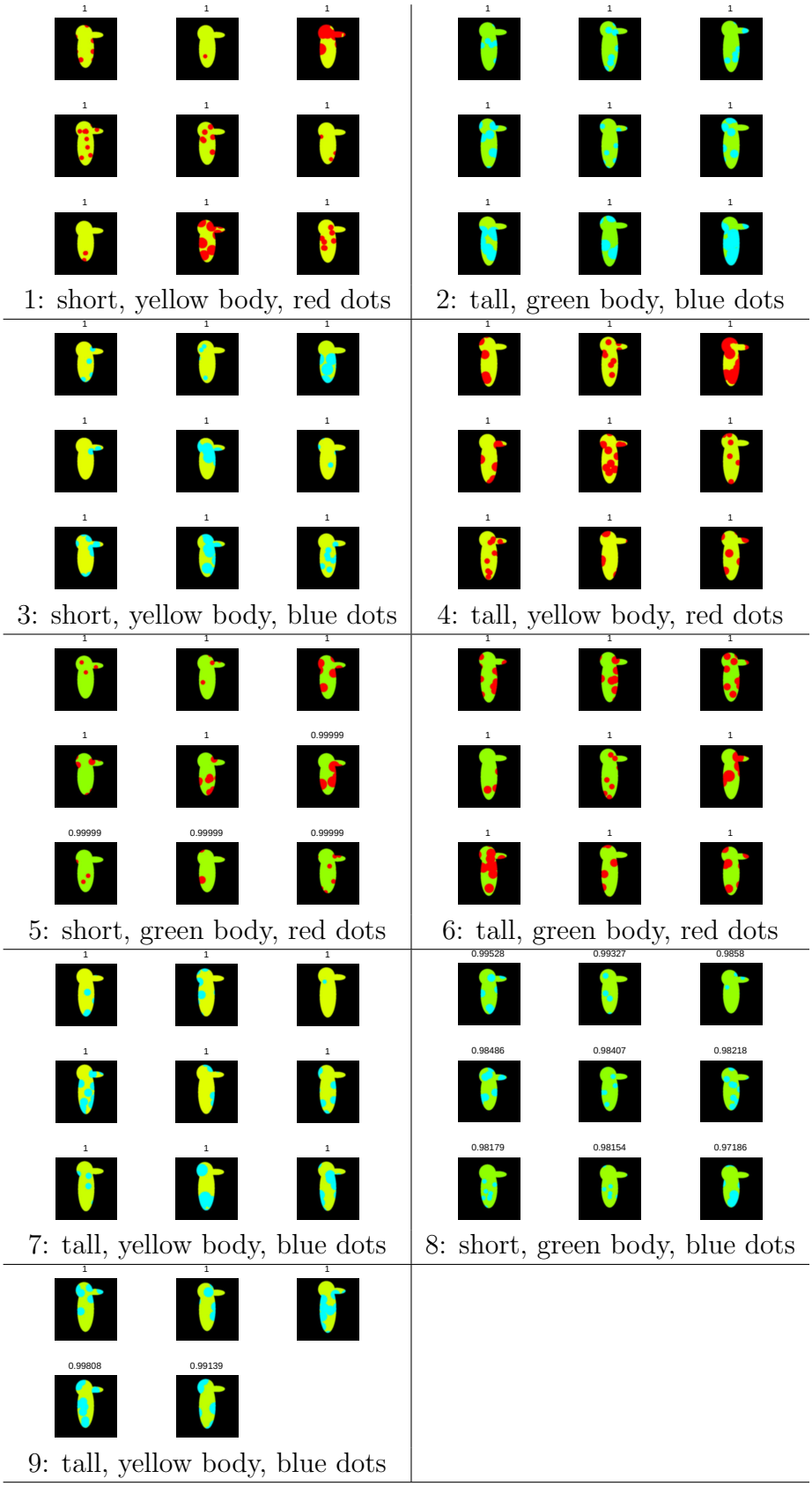


Figure 7.4: High confidence Greeble cluster examples. Each cluster may be described as a combination of three Greeble attributes: height, body color, and dot color.

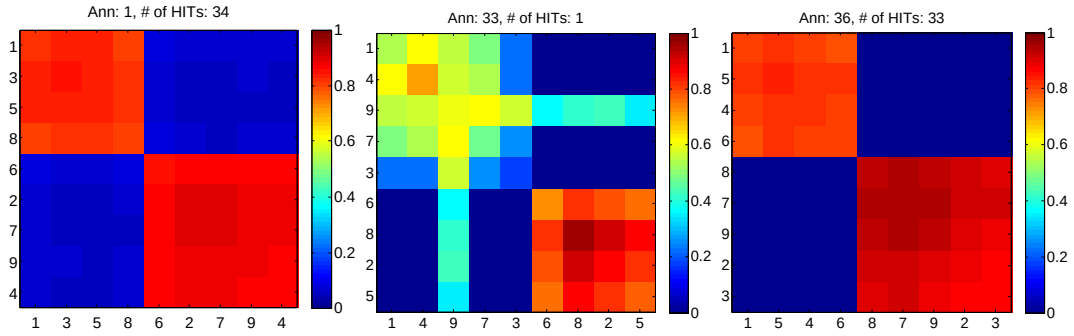


Figure 7.5: Confusion matrices $\hat{\mathbf{C}}_j$ for three types of workers in the Greeble experiment. Cluster indices were sorted to yield approximate block diagonal structure. Left: Worker cued to categorize based on Greeble height. Center: Worker cued to categorize based on Greeble body color. Only a single HIT was performed by this worker, and there is greater uncertainty about his categorization tendencies. Right: Some workers spontaneously decided to cluster based on the color of the Greeble's dots.

by mixture model clustering the resultant mean embedding $\boldsymbol{\mu}_i^x$. We find that a number of sensible categories emerge, which capture the basic activity (flying versus standing) and viewpoint orientation (left, right, center) of the birds. In addition, the fourth platonic cluster captures the majority of images whose pose is ambiguous.

Figure 7.8 shows example confusion matrices $\hat{\mathbf{C}}_j$ for two workers in the experiment. See the caption for details. We find that the confusion matrices are valuable for interpreting worker behavior.

7.5.3 Scenes

Our final experiment involves $N = 1001$ images from the scene dataset used in [FFP05]. Workers were instructed to categorize the images according to the type of scene.

High confidence examples from platonic clusters are shown in Figures 7.9 and 7.10. We find that they correspond to reasonable categories. However, with the exception of Bedroom scenes (platonic cluster 7), the indoor scenes are grouped together into platonic cluster 1. There are in fact three ground truth categories (office, living room, and kitchen) present in this cluster.

We postulate that this is a result of the ambiguity of the image categorization

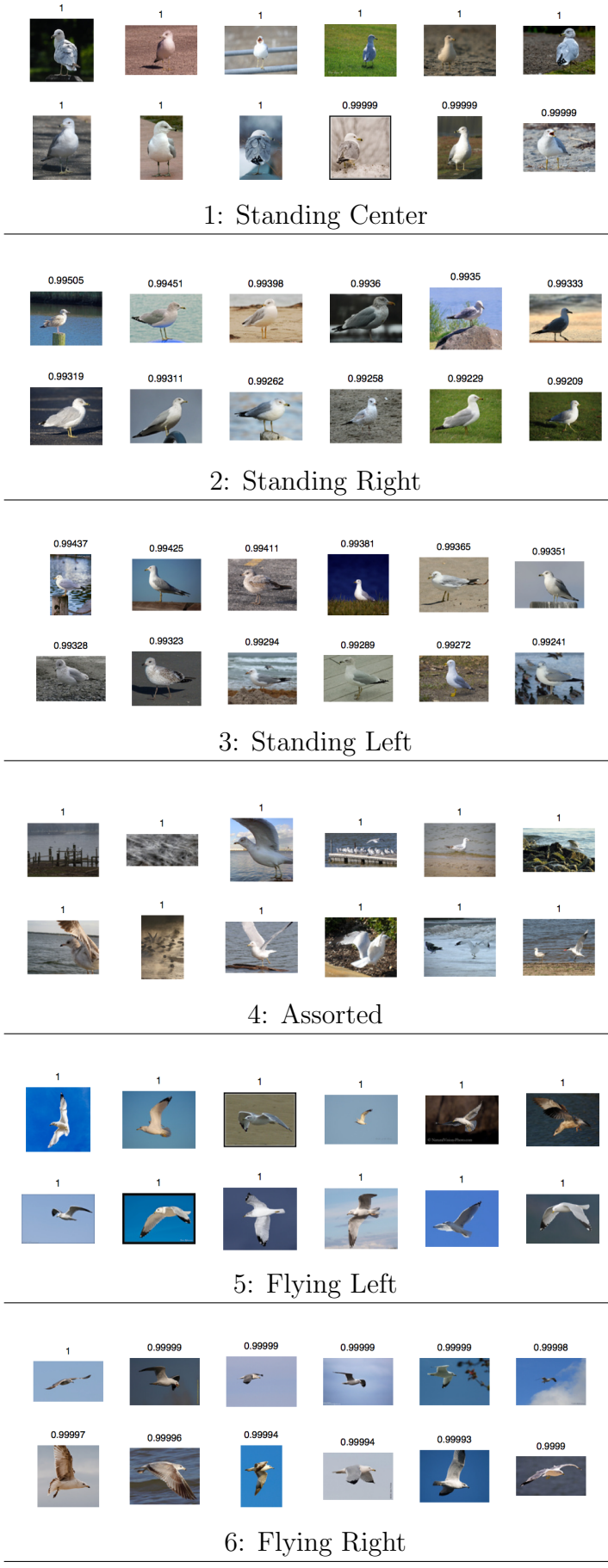
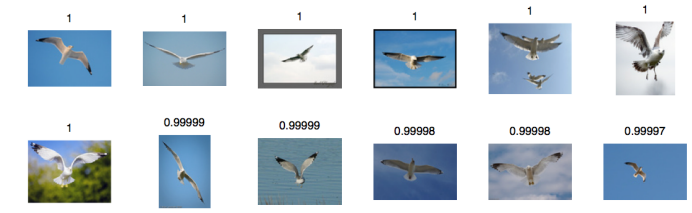


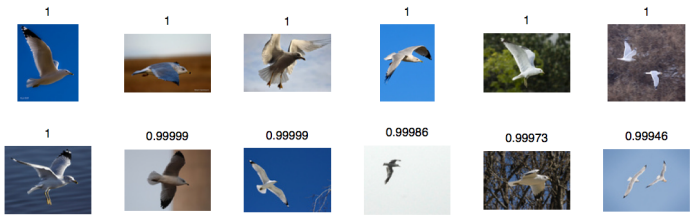
Figure 7.6: Bird Pose dataset: High confidence examples from inferred platonic clusters 1-6



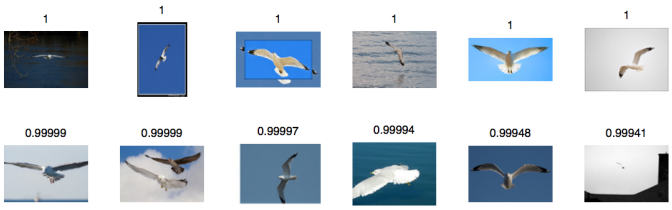
7: Flying Center

8: Head Right

9: Head Left



10: Flying Right



11: Flying Center

Figure 7.7: Bird Pose dataset: High confidence examples from inferred platonic clusters 7-11

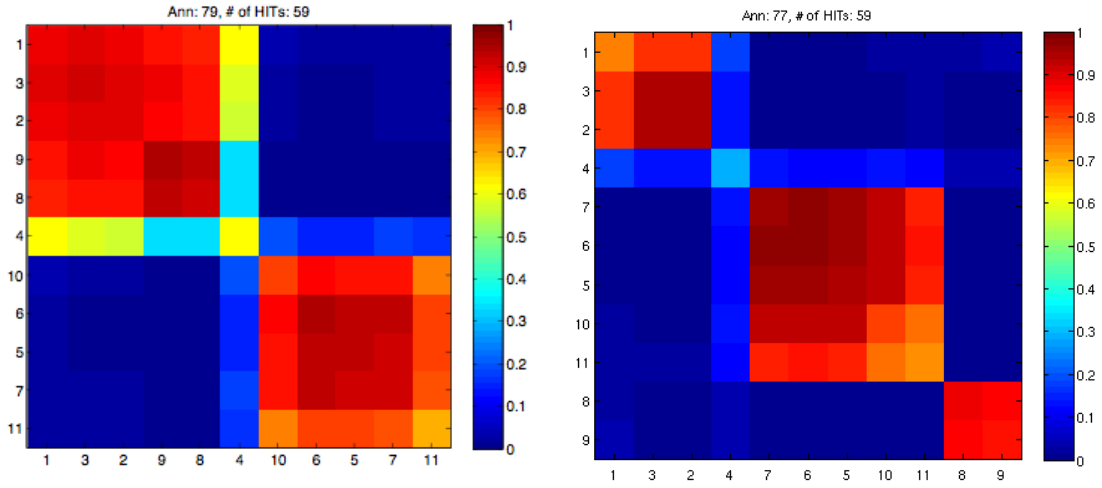


Figure 7.8: Confusion matrices $\hat{\mathbf{C}}_j$ for two workers in the Bird Pose experiment. Cluster indices were sorted to yield approximate block diagonal structure. Left: This worker divides images into two pose categories corresponding to birds in flight and birds on the ground. Right: This worker further divides birds on the ground into Standing pose and close up Head images. Note that the workers show uncertainty in grouping platonic cluster 4, which contains assorted images with poorly defined pose.

problem: there are many reasonable ways to categorize these images. The categorization behavior of workers is likely influenced by the implicit context of the task. For example, if a worker is presented with a HIT consisting of a few indoor scenes and a number of outdoor scenes, it is quite reasonable to use indoor versus outdoor as a discriminating factor. However, given a collection of purely indoor scenes in a HIT, it is likely that workers would draw finer distinctions that discriminate among indoor scene types such as living room and office.

We believe that intelligent sampling methods (mentioned in Section 7.4) may be used to counteract the effects of context. A "coarse-to-fine" scheme could be implemented in which workers first perform HITs according to the random sampling scheme given in 7.4, which would yield a coarse platonic clustering. Then, HITs are constructed which are composed of only members of a single (coarse) cluster, which would yield a refinement of the cluster into subcategories. This divisive clustering could be continued until a stopping criterion is met.

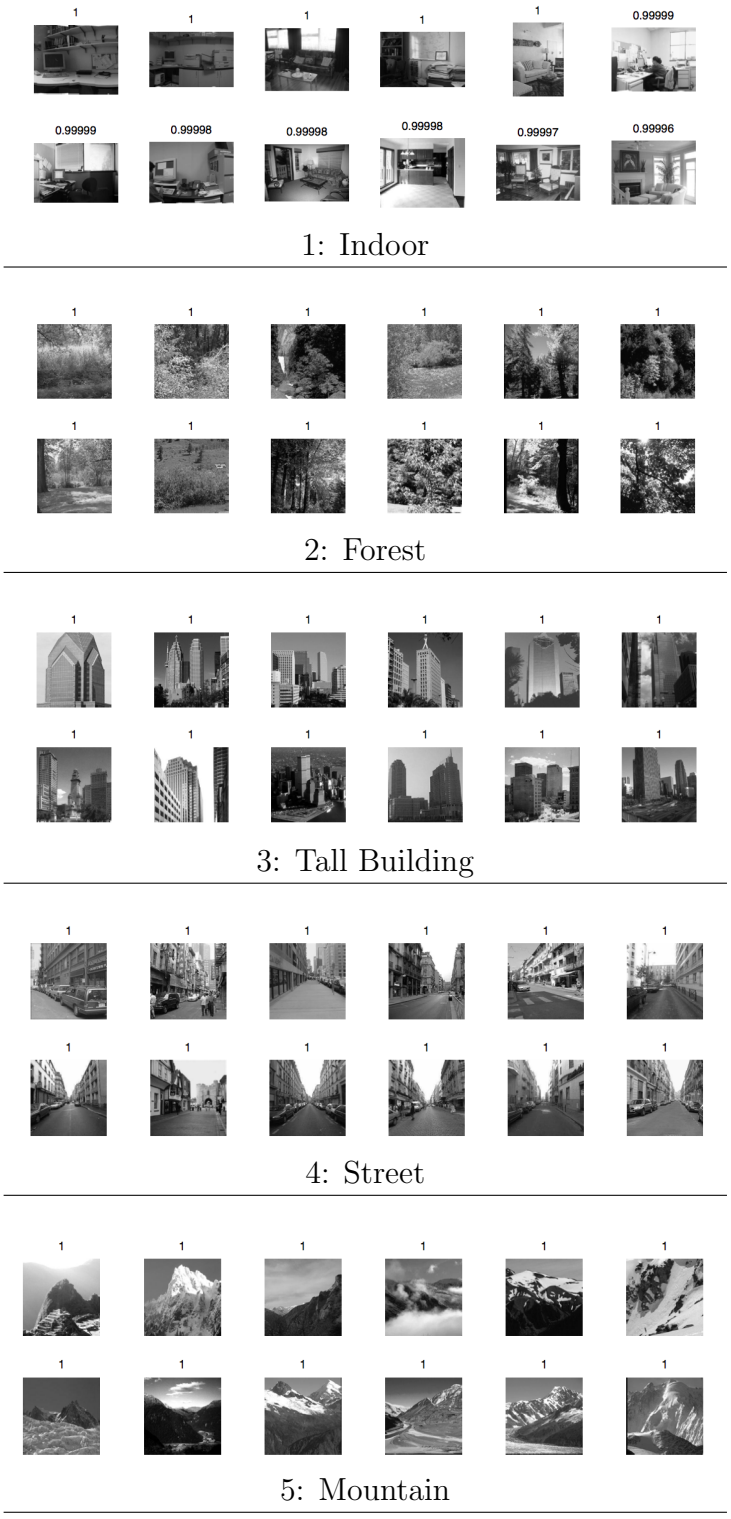
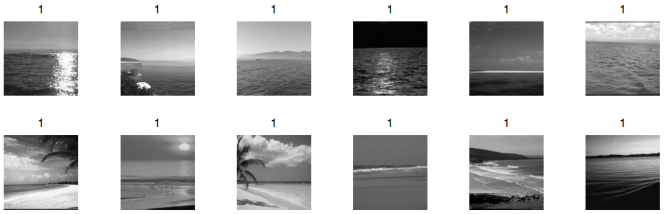


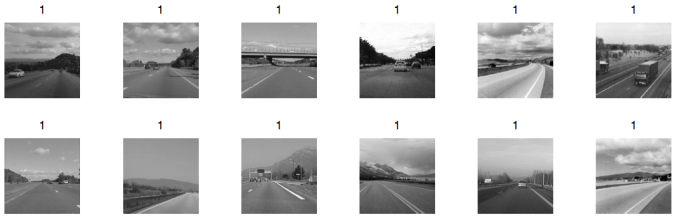
Figure 7.9: Scene dataset: High confidence examples from inferred platonic clusters 1-5

6: Suburb

7: Bedroom



8: Coast



9: Highway

10: Assorted

Figure 7.10: Scene dataset: High confidence examples from inferred platonic clusters 6-10

7.6 Discussion

We have defined the problem of Crowd Clustering: using crowdsourcing to categorize and analyze large collections of human-intepretable patterns. We have proposed two machine learning methods that may be used to aggregate clusterings of subsets of the data. In distinction from consensus clustering methods, ours may be used to model the multiple views or schools of thought among the crowd of workers.

Our experimental evaluation indicates that the methods are promising, and research on this topic is ongoing. Future work includes development of adaptive methods for selecting data items for HITs, experimental evaluation on large scale datasets where we expect a large number of platonic clusters, and quantitative experimental comparisons of the alternative Crowd Clustering and consensus clustering methods with respect to the item sampling redundancy factor V (see Section 7.4).

7.7 Appendix

The intermediate quantities necessary to compute the updates for $\boldsymbol{\mu}_i^x$ and $\boldsymbol{\sigma}_i^x$ are

$$
\begin{aligned}
[\mathbf{U}_i]_{d_1 d_2} = \sigma_{id_1}^x \Big(& \sum_{t:a_t=i} 2\lambda(\Delta_t) \Big[E_q\{\mathbf{W}_{j_t} \mathbf{x}_{b_t} \mathbf{x}_{b_t}^T \mathbf{W}_{j_t}\} \Big]_{d_1 d_2} \\
& + \sum_{t:b_t=i} 2\lambda(\Delta_t) \Big[E_q\{\mathbf{W}_{j_t} \mathbf{x}_{a_t} \mathbf{x}_{a_t}^T \mathbf{W}_{j_t}\} \Big]_{d_1 d_2} \Big) \qquad (7.7.1) \\
\mathbf{v}_i = \boldsymbol{\sigma}_i^x \circ \Big(& \sum_{t:a_t=i} \big(l_t/2 + 2\lambda(\Delta_t)\mu_{j_t}^\tau \big) \boldsymbol{\mu}_{j_t}^w \boldsymbol{\mu}_{b_t}^x \Big) \\
& + \sum_{t:b_t=i} \big(l_t/2 + 2\lambda(\Delta_t)\mu_{j_t}^\tau \big) \boldsymbol{\mu}_{j_t}^w \boldsymbol{\mu}_{a_t}^x \big) \Big) \\
E_q\{\mathbf{W}_j \mathbf{x}_i \mathbf{x}_i^T \mathbf{W}_j\} = & \boldsymbol{\mu}_j^w E_q\{\mathbf{x}_i \mathbf{x}_i^T\} \boldsymbol{\mu}_j^w + E_q\{\mathbf{x}_i \mathbf{x}_i^T\} \circ \boldsymbol{\sigma}_j^w \circ (\mathbf{1} - \mathbf{I}) \\
E_q\{\mathbf{x}_i \mathbf{x}_i^T\} = & \boldsymbol{\mu}_i^x (\boldsymbol{\mu}_i^x)^T + \mathrm{diag}\{\boldsymbol{\sigma}_i^x\}
\end{aligned}
$$

An alternative way to write the activity A_t which incorporates the symmetry

constraint on $\mathbf{W}_j$ is

$$A_t = \text{vecp}\{\mathbf{W}_{j_t}\}^T \text{vecp}\{\mathbf{y}^{a_t b_t}\} + \tau_{j_t}$$

where $\mathbf{y}^{a_t b_t} = \mathbf{x}_{a_t}\mathbf{x}_{b_t}^T + \mathbf{x}_{b_t}\mathbf{x}_{a_t}^T \circ (\mathbf{1} - \mathbf{I})$. We make use of this form of the activity when deriving updates for $\boldsymbol{\mu}_j^w$ and $\boldsymbol{\sigma}_j^w$. The intermediate terms required are:

$$\begin{aligned}
\mathbf{c}_j &= \text{vecp}\Big\{\boldsymbol{\sigma}_j^w \circ \Big(\sum_{t:j_t=j} (l_t/2 + 2\lambda(\Delta_t)\mu_j^\tau) E_q\{\mathbf{y}^{a_t b_t}\}\Big)\Big\} \qquad (7.7.2)\\
E_q\{\mathbf{y}^{ab}\} &= \boldsymbol{\mu}_a^x \boldsymbol{\mu}_b^x + \boldsymbol{\mu}_b^x \boldsymbol{\mu}_a^x \circ (\mathbf{1} - \mathbf{I})\\
\mathbf{B}_j &= \text{matp}\{\mathbf{T}_j\}\\
[\mathbf{T}_j]_{d_1 d_2 d_3 d_4} &= \sigma_{j d_1 d_2}^w \sum_{t:j_t=j} 2\lambda(\Delta_t)\big[E_q\{\mathbf{Y}^{a_t b_t}\}\big]_{d_1 d_2 d_3 d_4}\\
[E_q\{\mathbf{Y}^{ab}\}]_{d_1 d_2 d_3 d_4} &= (1 - \delta_{d_1 d_2} - \delta_{d_3 d_4} + \delta_{d_1 d_2}\delta_{d_3 d_4})\big[E_q\{\mathbf{x}_a\mathbf{x}_a^T\} \otimes E_q\{\mathbf{x}_b\mathbf{x}_b^T\}\big]_{d_1 d_3 d_2 d_4}\\
&\quad + (1 - \delta_{d_1 d_2})\big[E_q\{\mathbf{x}_a\mathbf{x}_a^T\} \otimes E_q\{\mathbf{x}_b\mathbf{x}_b^T\}\big]_{d_1 d_4 d_2 d_3}\\
&\quad + (1 - \delta_{d_3 d_4})\big[E_q\{\mathbf{x}_a\mathbf{x}_a^T\} \otimes E_q\{\mathbf{x}_b\mathbf{x}_b^T\}\big]_{d_2 d_3 d_1 d_4}\\
&\quad + \big[E_q\{\mathbf{x}_a\mathbf{x}_a^T\} \otimes E_q\{\mathbf{x}_b\mathbf{x}_b^T\}\big]_{d_2 d_4 d_1 d_3}
\end{aligned}$$

where $\mathbf{B}_j = \text{matp}\{\mathbf{T}_j\}$ reorganizes the fourth-order tensor $[\mathbf{T}]_{d_1 d_2 d_3 d_4}$ into a matrix $[B]_{n_1 n_2}$ such that index n_1 corresponds to the ordering of d_1 and d_2 produced by $\text{vecp}\{\cdot\}$ and index n_2 corresponds to the ordering on d_3 and d_4 produced by $\text{vecp}\{\cdot\}$. $\otimes$ represents the tensor product, and δ_{ij} is the Kronecker delta. Finally, $\mathbf{Y}^{ab} = \mathbf{y}^{ab} \otimes \mathbf{y}^{ab}$ are the correlation terms associated with $\mathbf{y}^{ab}$.

The following expectation is required to update Δ_t:

$$\begin{aligned}
E_q\{A_t^2\} &= E_q\{(\mathbf{x}_{a_t}^T \mathbf{W}_{j_t} \mathbf{x}_{b_t} + \tau_{j_t})^2\} \qquad (7.7.3)\\
&= \text{tr}\{E_q\{\mathbf{x}_{b_t}\mathbf{x}_{b_t}^T\}\boldsymbol{\mu}_{j_t}^w E_q\{\mathbf{x}_{a_t}\mathbf{x}_{a_t}^T\}\}\\
&\quad + \text{diag}\{E_q\{\mathbf{x}_{a_t}\mathbf{x}_{a_t}^T\}\}^T \boldsymbol{\sigma}_{j_t}^w \text{diag}\{E_q\{\mathbf{x}_{b_t}\mathbf{x}_{b_t}^T\}\}\\
&\quad + \mathbf{1}^T\mathbf{E}\mathbf{1} - \text{tr}\{\mathbf{E}\}
\end{aligned}$$

where $\mathbf{E} = E_q\{\mathbf{x}_{a_t}\mathbf{x}_{a_t}^T\} \circ E_q\{\mathbf{x}_{b_t}\mathbf{x}_{b_t}^T\} \circ \boldsymbol{\sigma}_{j_t}^w$, and $\text{tr}\{\cdot\}$ is the matrix trace operator.

Intermediate terms necessary to evaluate the cost function (Eq. 7.3.8) are as fol-

lows:

$$
\begin{aligned}
E_q \log p(X, W, \tau) = &-\frac{ND}{2}\log(2\pi\sigma_0^x) - \frac{1}{2\sigma_0^x}\sum_i (\boldsymbol{\mu}_i^x)^T\boldsymbol{\mu}_i^x + \mathbf{1}^T\boldsymbol{\sigma}_i^x \\
&- \frac{J}{2}\log(2\pi\sigma_0^\tau) - \frac{1}{2\sigma_0^\tau}\sum_j (\mu_j^\tau)^2 + \sigma_j^\tau \\
&- \frac{J(D^2+D)}{4}\log(2\pi\sigma_0^w) \\
&- \frac{1}{2\sigma_0^w}\sum_j (\mathrm{vecp}\{\boldsymbol{\mu}_j^w\})^T\mathrm{vecp}\{\boldsymbol{\mu}_j^w\} + \mathbf{1}^T\mathrm{vecp}\{\boldsymbol{\sigma}_j^w\} \\
\mathcal{H}\{q(X, W, \tau)\} = &\frac{1}{2}\sum_i\sum_d \log(2\pi e[\boldsymbol{\sigma}_i^x]_d) \\
&+ \frac{1}{2}\sum_j \big(\log(2\pi e\sigma_j^\tau) + \sum_{d_1\le d_2}\log(2\pi e[\boldsymbol{\sigma}_j^w]_{d_1 d_2})\big) \\
E_q\{A_t\} = &(\boldsymbol{\mu}_{a_t}^x)^T\boldsymbol{\mu}_{j_t}^w\boldsymbol{\mu}_{b_t}^x + \mu_{j_t}^\tau
\end{aligned}
\tag{7.7.4}
$$

The expression for $\hat{\Delta}_{k_1k_2j}$ (used for approximating the worker confusion matrix) is

$$\hat{\Delta}_{k_1k_2j} = \sqrt{E\{A_{k_1k_2j}^2\}}$$

where

$$
\begin{aligned}
E\{A_{k_1k_2j}^2\} = &\,\mathrm{tr}\{(\boldsymbol{\mu}_{k_1}\boldsymbol{\mu}_{k_1}^T + \boldsymbol{\Sigma}_{k_1})\boldsymbol{\mu}_j^w(\boldsymbol{\mu}_{k_2}\boldsymbol{\mu}_{k_2}^T + \boldsymbol{\Sigma}_{k_2})\} \\
&+ \mathrm{diag}\{\boldsymbol{\mu}_{k_1}\boldsymbol{\mu}_{k_1}^T + \boldsymbol{\Sigma}_{k_1}\}^T\boldsymbol{\sigma}_j^w\mathrm{diag}\{\boldsymbol{\mu}_{k_2}\boldsymbol{\mu}_{k_2}^T + \boldsymbol{\Sigma}_{k_2}\} \\
&+ \mathbf{1}^T\mathbf{F}\mathbf{1} - \mathrm{tr}\{\mathbf{F}\}
\end{aligned}
\tag{7.7.5}
$$

where $\mathbf{F} = (\boldsymbol{\mu}_{k_1}\boldsymbol{\mu}_{k_1}^T + \boldsymbol{\Sigma}_{k_1}) \circ (\boldsymbol{\mu}_{k_2}\boldsymbol{\mu}_{k_2}^T + \boldsymbol{\Sigma}_{k_2}) \circ \boldsymbol{\sigma}_j^w$.

Bibliography

[Ant74] CE Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics— Institute of Mathematical Statistics*, 1974.

[Att99] Hagai Attias. A variational baysian framework for graphical models. In *NIPS*, pages 209–215, 1999.

[BFR98] Paul S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *KDD*, pages 9–15, 1998.

[BH07] Francis Bach and Zaïd Harchaoui. DIFFRAC: a discriminative and flexible framework for clustering. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.

[BHM92] John S. Bridle, Anthony J. R. Heading, and David J. C. MacKay. Unsupervised classifiers, mutual information and 'phantom targets'. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 1096–1101. Morgan Kaufmann Publishers, Inc., 1992.

[BJ05] David M. Blei and Michael I. Jordan. Variational inference for dirichlet process mixtures. *Journal of Bayesian Analysis*, 1(1):121–144, 2005.

[BNJ03] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[BS94] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley, 1994.

[CB] Adrian Corduneanu and Christopher M. Bishop. Plenary papers 27, variational Bayesian model selection for mixture distributions.

[CBK+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[CJ03] A. Corduneanu and T. Jaakkola. On information regularization. In *UAI*, 2003.

[CO02] Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[COP03] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. In *STOC*, pages 30–39, 2003.

[Cre91] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1991.

[CSZ06] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[CWK05] Y. Chen, J. Ze Wang, and R. Krovetz. CLUE: cluster-based retrieval of images by unsupervised learning. *IEEE Trans. Image Processing*, 14(8):1187–1201, 2005.

[CZ04] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation, September 2004.

[Das09] S. Dasgupta. Lecture notes on online clustering. Technical report, http://www-cse.ucsd.edu/∼dasgupta/291/lec6.pdf, 2009.

[DD03] P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *J. Mol. Biol.*, 330:771–783, Jul 2003.

[DF07] D. Dueck and B. J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *ICCV*, pages 1–8, 2007.

[DH00] P. Domingos and G. Hulten. Mining high-speed data streams. In *KDD*, 2000.

[DH01] P. Domingos and G. Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In *ICML*, 2001.

[DK08] A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *STOC*, 2008.

[DLPP06] Chris H. Q. Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *KDD*, pages 126–135. ACM, 2006.

[DS03] G. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. In *International Conference on Computer Vision*, pages 634–640, 2003.

[DWSP09] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *CVPR*, June 2009.

[Fea04] P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Journal of Statistics and Computing*, 14:11–21, 2004.

[Fei98] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, July 1998.

[Fer73] Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1973.

[FFFP04] L. Fei-Fei, R. Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach

tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*, 2004.

[FFP05] Li Fei-Fei and Pietro Perona. A Bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531. IEEE Computer Society, 2005.

[FMR08] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*. IEEE Computer Society, 2008.

[GB04] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2004.

[GD06] Kristen Grauman and Trevor Darrell. Unsupervised learning of categories from sets of partially matching image features. In *CVPR (1)*, pages 19–25, 2006.

[GHP07] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[GMM+03] Guha, Meyerson, Mishra, Motwani, and O'Callaghan. Clustering data streams: Theory and practice. *IEEE TKDE*, 15, 2003.

[GMT07] Gionis, Mannila, and Tsaparas. Clustering aggregation. In *ACM Transactions on Knowledge Discovery from Data*, volume 1. 2007.

[GZK05] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *SIGMOD Record*, 34(2):18–26, June 2005.

[HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[JJ96] Tommi S. Jaakkola and Michael I. Jordan. A variational approach to Bayesian logistic regression models and their extensions, August 13 1996.

[JMJ99] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *NIPS*, 1999.

[JT05] Frédéric Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, pages 604–610. IEEE Computer Society, 2005.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. Wiley, 1990.

[Kru64] J. B. Kruskal. Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *PSym*, 29:1–29, 1964.

[KWV07] Kenichi Kurihara, Max Welling, and Nikos Vlassis. Accelerated variational dirichlet process mixtures. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[LCB+04] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[LDJ07] Tao Li, Chris H. Q. Ding, and Michael I. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *ICDM*, pages 577–582. IEEE Computer Society, 2007.

[LM99] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume II, pages 1010–1017, Los Alamitos, CA, September 20–27 1999. IEEE.

[LN89] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSP06] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[Min00] T. Minka. Estimating a dirichlet distribution. Technical report, 2000.

[MM94] Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *NIPS*, 1994.

[MN98] Andrew McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. 1998.

[MTMG03] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52(1–2):91–118, 2003.

[NH02] R. T. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Trans. Knowl. Data Eng*, 14(5):1003–1016, 2002.

[NJW01] A. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Math. Programming*, 14(1):265–294, December 1978.

[OPZ06] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR*, pages I: 3–10, 2006.

[Per10] P. Perona. Vision of a visipedia. *Proceedings of the IEEE*, 98(8):1526 –1534, Aug. 2010.

[Pia01] Justus H. Piater. *Visual feature learning.* PhD thesis, University of Massachusetts at Amherst, 2001.

[RW06] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning.* Adaptive Computation and Machine Learning. The MIT Press, 2006.

[Sat01] Masaaki Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[SATB05] N. Slonim, G. S. Atwal, G. Tkacik, and W. Bialek. Information-based clustering. *Proc Natl Acad Sci U S A*, 102(51):18297–18302, December 2005.

[SB00] Alex J. Smola and Peter L. Bartlett. Sparse greedy Gaussian process regression. In *NIPS*, pages 619–625. MIT Press, 2000.

[SB10] N. Shervashidze and K. M. Borgwardt. Fast subtree kernels on graphs. In *NIPS*, 2010.

[See04] M. Seeger. Greedy forward selection in the informative vector machine. Technical report, University of California at Berkeley, 2004.

[Set94] J Sethuraman. A constructive definition of dirichlet priors. *Statist. Sinica*, 4:639–650, 1994.

[SF08] A. Sorokin and D. A. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *Internet Vision*, pages 1–8, 2008.

[SG02] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[SG08] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, pages 1577–1584, 2008.

[SMS99] Alex J. Smola, Olvi L. Mangasarian, and Bernhard Scholkopf. Sparse kernel feature analysis, 1999.

[SOM06] Pawan Sinha, Yuri Ostrovsky, and Ethan Meyers. Parsing visual scenes via dynamic cues. *Journal of Vision*, 6(6):95, 2006.

[SRE+05] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their localization in images. In *ICCV*, pages 370–377, 2005.

[SSGB07] Le Song, Alex Smola, Arthur Gretton, and Karsten M. Borgwardt. A dependence maximization view of clustering. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 815–822. ACM, 2007.

[STC04] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[SWL03] M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *AISTATS*, 2003.

[TC01] Simon Tong and Edward Y. Chang. Support vector machine active learning for image retrieval. In *ACM Multimedia*, pages 107–118, 2001.

[Teh06] Y. W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *ACL*, 2006.

[TFF08] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE PAMI*, 30(11):1958–1970, November 2008.

[TJBB06] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, December 2006.

[TKW08] Y. W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for HDP. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[TM95] Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1–2):25–46, 1995.

[TPB00] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *CoRR*, physics/0004057, 2000.

[UNGH99] N. Ueda, R. Nakano, Z. Gharamani, and G. Hinton. Smem algorithm for mixture models, 1999.

[VJ01] Paul A. Viola and Michael J. Jones. Robust real-time face detection. In *ICCV*, page 747, 2001.

[VL99] Nuno Vasconcelos and Andrew Lippman. Learning mixture hierarchies. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 606–612. MIT Press, 1999.

[VNU03] Michel Vidal-Naquet and Shimon Ullman. Object recognition with informative features and linear classification. In *ICCV*, pages 281–288. IEEE Computer Society, 2003.

[VNV03] J. Verbeek, J. Nunnink, and N. Vlassis. Accelerated variants of the em algorithm for Gaussian mixtures. Technical report, University of Amsterdam, 2003.

[VNV06] Jakob J. Verbeek, Jan Nunnink, and Nikos A. Vlassis. Accelerated em-based clustering of large data sets. *Data Min. Knowl. Discov.*, 13(3):291–307, 2006.

[VR07] Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, pages 1–8. IEEE, 2007.

[WBB05] Jason Weston, Antoine Bordes, and Léon Bottou. Online (and offline) on an even tighter budget. In *AISTATS*, pages 413–420, 2005.

[WBBP10] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Neural Information Processing Systems Conference (NIPS)*, 2010.

[WBM+10] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[WK06] Nikil Wale and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *ICDM*, pages 678–689, 2006.

[WWP00] Markus Weber, Max Welling, and Pietro Perona. Towards automatic discovery of object categories. In *CVPR*, 2000.

[XNJR02] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. Distance metric learning with application to clustering with side-information. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 505–512. MIT Press, 2002.

[XS05] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *AAAI*, 2005.

[Zit08] J. Zittrain. Ubiquitous human computing. *Royal Society of London Philosophical Transactions Series A*, 366:3813–3821, October 2008.

[ZTK07] K. Zhang, I. W. Tsang, and J. T. Kwok. Maximum margin clustering made practical. In *ICML*, 2007.

[Zuc00] W. Zucchini. An Introduction to Model Selection. *Journal of Mathematical Psychology*, 44(1):41–61, March 2000.

[ZWFM06] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava, editors, *SDM*. SIAM, 2006.