



Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2014-011

May 26, 2014

Latent Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification

Been Kim, Cynthia Rudin, and Julie Shah

Latent Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification

Been Kim

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
beenkim@csail.mit.edu

Cynthia Rudin

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
rudin@mit.edu

Julie Shah

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
julie_a_shah@csail.mit.edu

Abstract

We present a general framework for Bayesian case-based reasoning and prototype classification and clustering — Latent Case Model (LCM). LCM learns the most representative *prototype* observations of a dataset by performing joint inference on cluster prototypes and features. Simultaneously, LCM pursues sparsity by learning *subspaces*, the sets of few features that play important roles in characterizing the prototypes. The prototype and subspace representation preserves interpretability in high dimensional data. We validate the approach preserves classification accuracy on standard data sets, and verify through human subject experiments that the output of LCM produces statistically significant improvements in participants' performance on a task requiring an understanding of clusters within a dataset.

1 Introduction

People like to look at examples. Marketers show us examples of people we might want to emulate in order to lure us into a purchase. We might ignore Amazon.com's recommendations and look instead at an Amazon customer's Listmania to find an example of a customer like us. We might ignore medical charts computed from lots of patients in favor of medical blogs where we can get examples of real patients' experiences.

Numerous studies demonstrate that this type of exemplar-based reasoning, involving various forms of matching and prototyping, is also fundamental to our most effective strategies for tactical decision-making ([31, 29, 8, 10, 23, 27, 32]). For example, naturalistic studies show that skilled decision-makers in the fire service use *recognition-primed decision making*, in which new situations are matched to typical cases where certain actions are appropriate and usually successful [23]. To assist humans in leveraging large data sources to make better decisions, we desire that machine learning algorithms provide their outputs in forms that are easily incorporated into the human decision-making process.

These studies of human decision-making and cognition provided the key inspiration for artificial intelligence Case-Based Reasoning (CBR) approaches [2, 35, 36, 24]. CBR relied on the idea that a new situation can be well-represented by the summarized experience of previously solved problems [24]. CBR has been applied to important real world applications [22, 26, 5], but is fundamentally limited in that it does not learn the underlying structure of data in an unsupervised fashion, and may not scale to datasets with high-dimensional feature spaces (as discussed in [37]).

In this work we introduce a new Bayesian model, called the Latent Case Model (LCM), for prototype clustering and subspace learning. The prototype is the exemplar that is most representative of the cluster. The subspace representation is a powerful output of the model because we do not need (nor would we want) the best exemplar to be similar to the current situation in all possible ways: for instance, a moviegoer who likes the same horror films as we do might be useful for identifying good horror films, regardless of their cartoon preferences. We model the underlying data as a mixture model and infer set of features that are important within each cluster (i.e., subspace). This type of model can help to bridge the gap between machine learning methods and humans, who use examples as a fundamental part of their decision-making strategy.

We show that LCM produces comparable or better prediction accuracy than prior art for standard datasets. We also verify through human subject experiments that the prototypes and subspaces present as meaningful feedback for characterizing the important aspects of a dataset. The exemplar-based output of LCM produced statistically significant improvements in participants' performance on a task requiring an understanding of clusters within a dataset, as compared to outputs produced by prior art [7].

2 Background and Related Work

People organize and interpret information through exemplar-based reasoning, particularly when they are solving problems ([31, 29, 8, 10, 23, 27, 32]). AI Cased-Based Reasoning approaches are motivated by this insight, and provide examples cases along with the machine-learned solution. Studies show that example cases significantly improve user confidence in the solutions, as compared to showing only the solution or by also displaying a rule that was used in finding the solution [12]. CBR requires solutions (i.e. labels) for previous cases and therefore does not learn the underlying structure of the data in an unsupervised fashion. Maintaining transparency in complex situations also remains a challenge [37]. CBR models designed explicitly to produce explanations [1] rely on backward chaining the causal relation from a solution, which does not scale as the complexity increases. The cognitive load of the user also increases with the complexity of similarity measure for comparing cases [16]. Other CBR models for explanations require the model to be manually crafted in advance by experts [30].

Alternatively, mixture models are a powerful tool for discovering cluster distributions in an unsupervised fashion. However, this approach does not provide intuitive explanations for the learned clusters, as pointed out in [9]. Sparse topic models are designed to improve interpretability by reducing the number of words per topic [3, 39, 14]. However, using the number of features as a proxy for interpretability is problematic, as the metric is independent of the contents of the model or the data [16]. Explanations produced by mixture models are typically presented as distributions over word features. Even users with technical expertise in machine learning may have a hard time interpreting this output, especially when the cluster distribution is over a large number of features [9].

Our approach, the Latent Case Model (LCM) simultaneously performs unsupervised clustering, and learns the most representative case (i.e., prototype) and the important features (i.e., subspaces). In this work we use the standard discrete mixture model to learn the underlying structure, and this can be generalized to other unsupervised clustering techniques. The LCM preserves the power of CBR in generating interpretable output. In our view, there are at least three widely known types of interpretable models: sparse linear classifiers (e.g., the lasso [38], also see [9]), discretization methods such as decision trees and decision lists (e.g., [13, 34, 3, 39, 14]), and prototype or case-based classifiers (e.g., nearest neighbors [11] or a supervised optimization-based method [6]). Extensive review on interpretable classification methods is found from [16]. LCM aims at the third type, but through unsupervised generative mechanisms for explaining clusters, rather than supervised optimization-based approaches or by focusing on local points in the data space [4].

3 Latent Case Model

We formally describe the Latent Case Model (LCM) in this section. Intuitively, the goal is to have each observation be generated from the important pieces of related prototypes. For instance, it might generate horror and action movie suggestions from one person with similar tastes in those genres, and cartoon suggestions from some else's example.

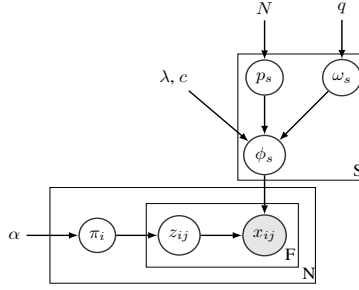


Figure 1: Latent Case Model

Our model starts with a standard discrete mixture model [19, 7] to represent the underlying structure of the observations. LCM augments the standard mixture model with *prototypes* and *subspace feature indicators* that characterize the clusters. We show in Sections 4.3 that prototypes and subspace feature indicators improve human interpretability as compared to the standard mixture model output. The graphical model for LCM is shown in Figure 1.

We start with N observations denoted by $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$, and each x_i is represented as a random mixture over clusters. There are S clusters, where S is assumed to be known in advance. (This assumption can easily be relaxed through extension to a non-parametric mixture model if desired.) Vector π_i are the mixture weights over these clusters for the i^{th} observation x_i , $\pi_i \in \mathbb{R}_+^S$. Each observation has P features, and we denote the j^{th} feature of the i^{th} observation as x_{ij} . Each feature j of the observation x_i comes from one of the clusters, the index of the cluster for x_{ij} is denoted by z_{ij} , and the full set of cluster assignments for observation-feature pairs is denoted by \mathbf{z} . Each z_{ij} takes on a value that is a cluster index between 1 and S . Hyperparameters q , λ and α are assumed to be fixed.

The explanatory power of LCM is in how the clusters are characterized. While a standard mixture model assumes that each cluster take the form of a predefined parametric distribution (e.g., normal), LCM characterizes each cluster by a *prototype*, p_s , and *subspace feature indicator*, ω_s . Intuitively, the *subspace feature indicator* selects only a few features that play an important role in identifying the cluster and prototype (hence LCM clusters are *subspace clusters*). Below we intuitively define these latent variables.

Prototype, p_s : The prototype p_s for subspace s is defined as one observation in \mathbf{x} that maximizes $p(p_s|\omega_s, \mathbf{z}, \mathbf{x})$, where the probability density and ω_s are defined below. Our notation for element j of p_s is p_{sj} , and since p_s is a prototype, it is equal to one of the observations, so $p_{sj} = x_{ij}$ for some i . Note that there may exist more than one prototype per subspace cluster; in this case one is arbitrarily chosen. Intuitively, the prototype is the “quintessential” observation that best represents the subspace cluster.

Subspace feature indicator ω_s : Intuitively, ω_s ‘turns on’ the features that are important in characterizing subspace cluster s and choosing the prototype, p_s . Here $\omega_s \in \{0, 1\}^P$ is an indicator variable that is 1 on the subset of features that maximizes $p(\omega_s|p_s, \mathbf{z}, \mathbf{x})$, where the probability for ω_s is defined below. Here ω_s is a binary vector of size P , where each element is an indicator for whether feature j belongs to subspace s .

The generative process for the Latent Case Model (LCM) is as follows. First, we start by generating the subspace clusters. A subspace cluster can be fully described by three components 1) a prototype, p_s , which is generated by sampling uniformly over all observations, $1 \dots N$; 2) a feature indicator vector, ω_s , that indicates important features for that subspace cluster, where each element of the feature indicator (ω_{sj}) is generated according to a Bernoulli distribution with hyperparameter q ; 3) the distribution of feature outcomes for each feature, ϕ_s for subspace s that we now describe.

Distribution of feature outcomes ϕ_s for cluster s : Here ϕ_s is a data structure where each “row” ϕ_{sj} is a discrete probability distribution of possible outcomes for feature j . Explicitly, ϕ_{sj} is a vector of length V_j , where V_j is the number of possible outcomes of feature j . Let us define Θ to be a vector of the possible outcomes of feature j (e.g., for feature ‘color’, $\Theta = [\text{red}, \text{blue}, \text{yellow}]$), where Θ_v represents a particular outcome for the feature (e.g., $\Theta_v = \text{blue}$). We will generate ϕ_s so that it

mostly takes outcomes from the prototype p_s for the important dimensions of the subspace cluster. We do this by considering the vector g , indexed by possible outcomes v , as follows:

$$g_{p_{sj}, \omega_{sj}, \lambda}(v) = \lambda(1 + c\mathbb{1}_{[w_{sj}=1 \text{ and } p_{sj}=\Theta_v]})$$

where c and λ are constant hyperparameters that indicate how much we will copy the prototype in order to generate the observations. The distribution of feature outcomes will be determined by g through $\phi_{sj} \sim \text{Dirichlet}(g_{p_{sj}, \omega_{sj}, \lambda})$. To explain at the intuitive level, start by considering the irrelevant dimensions j in subspace s , which have $w_{sj} = 0$. In that case, ϕ_{sj} will look like a uniform distribution over all possible outcomes for features j ; the feature values for the unimportant dimensions are generated arbitrarily according to the prior. Now consider relevant dimensions where $w_{sj} = 1$. In that case, ϕ_{sj} will generally take on a larger value $\lambda + c$ for the feature value that prototype p_s has on feature j , which is called Θ_v . All of the other possible outcomes are taken with lower probability λ . This means we will be more likely to choose the outcome Θ_v that agrees with the prototype p_s .

In the extreme case where c is very large, we can think of copying the cluster’s prototype directly within the cluster’s relevant subspace, and assigning the rest of the feature values randomly. An observation is then a mix of different prototypes, where we take the important pieces of each prototype. Specifically, we generate the observations as follows. For element j of observation i , we select a cluster, according to its subspace cluster distribution π_i . Here, π_i is generated according to a Dirichlet distribution, parameterized by hyperparameter α . From there, to obtain the cluster index z_{ij} for each x_{ij} , we sample from a multinomial distribution with parameters π_i . Finally, each feature for an observation, x_{ij} is sampled from the feature distribution of the assigned subspace cluster ($\phi_{z_{ij}}$). Note that this part of our model is the standard mixture model formulation. LDA also starts with a standard mixture model, though our feature values lie in a discrete set that is not necessarily binary. Here is the full model, with hyper parameters c , λ , q , and α :

$$\begin{aligned} \omega_{sj} &\sim \text{Bernoulli}(q) \quad \forall s, j & p_s &\sim \text{Uniform}(1, N) \quad \forall s \\ \phi_{sj} &\sim \text{Dirichlet}(g_{p_{sj}, \omega_{sj}, \lambda}) \quad \forall s, j \text{ where } g_{p_{sj}, \omega_{sj}, \lambda}(v) = \lambda(1 + c\mathbb{1}_{[w_{sj}=1 \text{ and } p_{sj}=\Theta_v]}) \\ \pi_i &\sim \text{Dirichlet}(\alpha) \quad \forall i & z_{ij} &\sim \text{Multinomial}(\pi_i) \quad \forall i, j & x_{ij} &\sim \text{Multinomial}(\phi_{z_{ij}}) \quad \forall i, j. \end{aligned}$$

Our model can be readily extended to different similarity measures, such as standard kernel methods or domain specific similarity measures, by modifying the function g . For example, we can use the least squares loss i.e., for fixed threshold ϵ , $g_{p_{sj}, \omega_{sj}, \lambda}(v) = \lambda(1 + c\mathbb{1}_{[w_{sj}=1 \text{ and } (p_{sj}-\Theta_v)^2 \leq \epsilon]})$, or more generally, $g_{p_{sj}, \omega_{sj}, \lambda}(v) = \lambda(1 + c\mathbb{1}_{[w_{sj}=1 \text{ and } \ell(p_{sj}, \Theta_v) \leq \epsilon]})$.

In terms of setting hyperparameters, there are natural settings for alpha (all entries being 1). This means there are three real-valued parameters to set, which can be set through cross validation, another layer of hierarchy with more diffuse hyperparameters, or plain intuition.

3.1 Motivating example

This section provides an illustrative example for prototypes, subspace feature indicator and subspace clusters using a dataset composed of a mixture of smiley faces. The feature set for the smiley face is composed of eyes and mouth types, shape and color. For the purpose of this example, assume that the ground truth is that there are three clusters, each of which has two features that are important for defining the cluster. In Table 1, we show the first cluster, whose subspace is defined by color (green) and shape (square) of the face, and the rest of the features are not important for defining the cluster. For the second cluster, color (orange) and eye shape define the subspace. We generated 240 smiley faces from LCM’s prior with $\alpha = 0.1$ for all entries, and $q = 0.5$, $\lambda = 2$ and $c = 50$.

We explain the way LCM works by contrasting it with Latent Dirichlet Allocation (LDA) [7], which presents its output in very different form. Table 1 shows the representation of clusters in LDA (middle column) and LCM (right column). This dataset is particularly simple, and we chose this comparison because the two most important features that both LDA and LCM learn are identical for each cluster; however, LDA does not learn prototypes and represents information differently. To convey cluster information using LDA (to define a topic), we need to record several probability distributions - one for each feature. For LCM, we need only to record a prototype (e.g., the green face in the top row, right column of the figure), and state which features were important for the cluster’s subspace (e.g., shape and color). For this reason, LCM is more succinct than LDA with what










	Data in assigned to cluster	LDA	LCM	
		Top 3 words and probabilities	Prototype	Subspaces
1		 0.26 0.23 0.12		shape and color are important.
2		 0.26 0.24 0.16		color and eye are important.
3		 0.35 0.27 0.15		eye and mouth are important.

Table 1: Mixture of smiley faces for LDA and LCM

information needs to be recorded to define the clusters. One could define a “special” constrained version of LDA whose topics all have uniform weights over a subset of features, and whose “word” distributions are centered around a particular value; this would require a similar amount of memory, however it loses information with respect to fact that LCM carries a full prototype with it for each cluster.

A major benefit of LCM over LDA is that the “words” in each topic (the choice of feature values) are coupled, and are not assumed to be independent; correlations can be controlled depending on the choice of parameters. The independence assumption of LDA can be very strong, and possibly crippling for many important applications; following our example of images, one could easily generate an image with eyes and a nose that are physically not possible to occur in the same person (perhaps overlapping). LCM can also generate this, but it would be very unlikely, as it would generally prefer to copy the important features from a prototype.

LCM performs joint inference on prototypes, subspace feature indicators and cluster labels for observations. This encourages the inference step to achieve solutions where clusters are better represented by prototypes. We will show that this has a benefit in terms of predictive accuracy in Section 4.1. We also show that LCM’s succinct representation is very effective in communicating the characteristics of clusters through a human subject experiment in Section 4.3

3.2 Inference: collapsed Gibbs sampling

We use collapsed Gibbs sampling to perform inference, as this has been observed to converge quickly, particularly in LDA mixture models [17]. We sample ω_{sj} and z_{ij} , and p_s , where ϕ and π are integrated out. Note that we can always recover ϕ by simply counting the number of feature values that are assigned to each subspace.

Integrating out ϕ and π gives the following expression for sampling z_{ij} .

$$p(z_{ij} = s | z_{i-\cdot j}, x, p, \omega, \alpha, \lambda) \propto \frac{\alpha/S + n_{(s,i,-j,\cdot)}}{\alpha + n} \times \frac{g(p_{sj}, \omega_{sj}, \lambda) + n_{(s,\cdot,j,x_{ij})}}{\sum_s g(p_{sj}, \omega_{sj}, \lambda) + n_{(s,\cdot,j,\cdot)}}, \quad (1)$$

where $n_{(s,i,j,v)} = \mathbb{1}(z_{ij} = s, x_{ij} = v)$. In other words, if the observation x_{ij} takes feature value v for feature j and is assigned to subspace cluster s , then $n_{(s,i,j,v)} = 1$, and 0 otherwise. $n_{(s,\cdot,j,v)}$ is the number of times that the j^{th} feature of an observation takes feature value v and that observation is assigned to subspace cluster s (i.e., $n_{(s,\cdot,j,v)} = \sum_i \mathbb{1}(z_{ij} = s, x_{ij} = v)$). We use $n_{(s,i,-j,v)}$ to denote a count that does not include the feature j . The derivation follows the standard collapsed Gibbs sampling for LDA mixture models [17].

Similarly, integrating out ϕ gives the following expression for sampling ω_{sj} .

$$p(\omega_{sj} = b|q, p_{sj}, \lambda, \phi, x, z, \alpha) \propto \begin{cases} q \times \frac{\mathbf{B}(g(p_{sj}, 1, \lambda) + n_{(s, \cdot, j, \cdot)})}{\mathbf{B}(g(p_{sj}, 1, \lambda))} & b = 1 \\ 1 - q \times \frac{\mathbf{B}(g(p_{sj}, 0, \lambda) + n_{(s, \cdot, j, \cdot)})}{\mathbf{B}(g(p_{sj}, 0, \lambda))} & b = 0 \end{cases} \quad (2)$$

where \mathbf{B} is the Beta function, and comes from integrating out ϕ variables, which are sampled from Dirichlet distributions.

4 Results

In this section we show that LCM produces comparable or better prediction accuracy than LDA for standard datasets. Next, we visually illustrate that the learned prototypes and subspaces present as meaningful feedback for characterizing the important aspects of the dataset. Finally, we verify through human subject experiments the benefit of LCM using a task requiring an understanding of clusters within a dataset. We show statistically significant improvements in objective measures of task performance using prototypes produced by LCM, compared to output of LDA.

4.1 LCM maintains prediction accuracy.

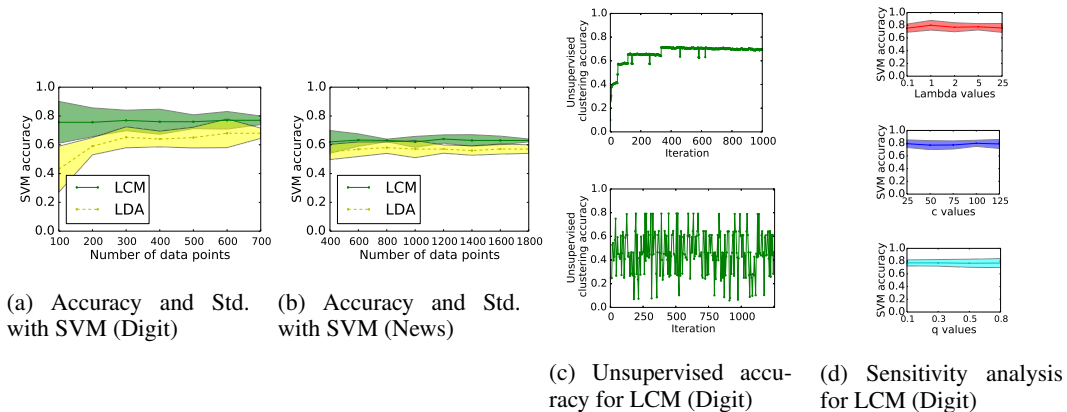


Figure 2: Prediction accuracy reported for the *Handwritten Digit* and *20 Newsgroups* datasets. (a) and (b) apply SVM for both LDA and LCM. (c) presents the unsupervised accuracy for LCM and (d) presents the sensitivity analysis for hyperparameters. Datasets were produced by randomly sampling 10 to 70 observations of each digit for the *Handwritten Digit* dataset, and 100 - 450 documents per document class for the *20 Newsgroups* dataset. The *Handwritten Digit* pixel values (range from 0 to 255) were rescaled into 7 bins (range from 0 to 6). Each 16 by 16 pixel picture was represented as a 1D vector of pixels values with length 256. Both LCM and LDA were randomly initialized with the same seed (half the labels were incorrect and randomly mixed), Number of iterations was set at 1000. $S = 4$ for *20 Newsgroups* and $S = 10$ for *Handwritten Digit*. $\alpha = 0.01$, $\lambda = 2$, $c = 50$, $q = 0.8$.

We show that the LCM output produces comparable or better prediction accuracy than LDA, which uses the same mixture model (Section 3) to learn the underlying structure but does not learn explanations (i.e., prototypes and subspaces). We validate this for two standard datasets, *Handwritten Digit* [20] and *20 Newsgroups* [25] datasets. We use the implementation of LDA available from [33], which uses the same inference technique (Gibbs sampling) as LCM.

Figure 2a and 2b show the ratio of correctly assigned cluster labels for LCM and LDA. Here the learned cluster labels (z) are provided as a set of features to support vector machine (SVM), as is often done in LDA literature for clustering [7]. The improved accuracy of LCM over LDA, as depicted in the figures, is explained in part that LCM captures the dependencies among features via prototypes, as described in Section 3. We also note that prediction accuracy using the full *20 Newsgroups* dataset acquired by LDA (accuracy: 0.68 ± 0.01) matches that reported previously for

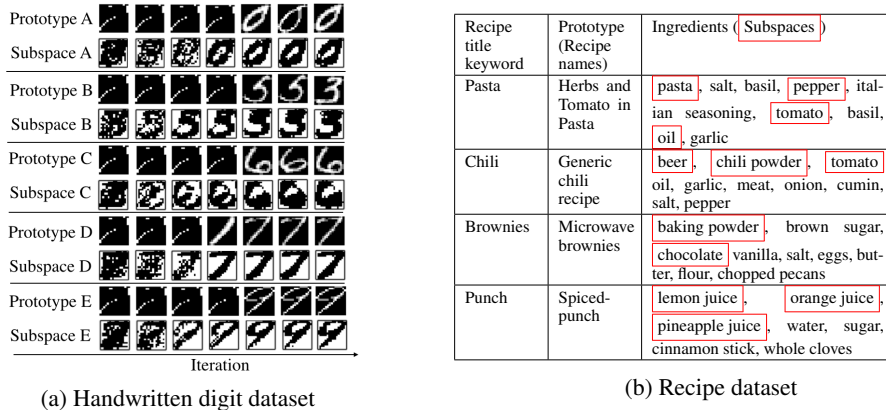


Figure 3: Learned prototypes and subspaces for the *Handwritten Digit* and *Recipe* datasets.

this dataset using a combined LDA and SVM approach [40], and that results for LDA on the full *Handwritten Digit* dataset (accuracy: 0.76 ± 0.017) is comparable to that produced by LCM with the subsampled dataset (70 samples per digit, accuracy: 0.77 ± 0.03).

Figure 2c shows the unsupervised clustering accuracy for LCM (no SVM). We can compute this measure for LCM because each cluster is characterized by a prototype, which is a particular data point that has a label in the given datasets. Note that this is not possible to do for LDA. We assume that all data points assigned to a cluster have the same label as cluster’s prototype. We also provide sensitivity analysis on the set of hyperparameters introduced to learn prototypes and subspaces (i.e., q , λ and c) in Figure 2d.

4.2 Learning subspaces

Figure 3a visually illustrates the learned prototypes and subspaces as a function of sampling iterations for the *Handwritten Digit* dataset. Figure 3b shows sets of ingredients learned as prototypes and subspaces for a standard dataset of cooking recipes ¹. One can see that the output of the LCM presents as meaningful feedback for characterizing the important aspects of the dataset. For example, in handwritten digit dataset, the subspaces learn the pixels that define the digit in the corresponding prototype.

Due to random initialization, early iterations in learning the subspaces first identify features that are common to observations randomly initialized in that subspace cluster. This is because ω_s assigns higher likelihood to features that are most common across observations within that cluster. For example, most of digits ‘agree’ (i.e., have the same zero pixel value) near the borders, thus these are the first areas that are refined as shown in Figure 3a. The sparsity of the subspaces can be customized by the hyperparameter q .

Note that LCM makes mistakes that are similar those reported using LDA for the *Handwritten Digit* dataset [4]. For example, the digits 3 and 5 (shown in the third row in Figure 3a) are usually assigned to the same subspace cluster, as they share many pixel values in similar locations. This effect is amplified since we represent digits as a flattened-out 1D vector rather than a 2D array. Performance can be improved using more sophisticated features (e.g., [18], [15] and [28]).

4.3 Verifying interpretability of LCM

Finally, we verified the benefit of LCM through human subject experiments using an experiment task that required an understanding of clusters within a dataset. The task required each participant to assign 16 recipes (described by a set of ingredients required to cook the recipe; recipe names and instructions were withheld) to one of a set of 4-6 cluster representations. This experiment approach is similar to those used previously to measure comprehensibility [21]. We chose a recipe data for

¹Available for computer cooking contest: <http://liris.cnrs.fr/ccc/ccc2014/>

New dish	Categories	Descriptions of unnamed categories																																																																																																
salt butter milk water vanilla raspberry ricotta orange nutmeg flour egg starch powder sugar brandy zest Your answer?	brownie cheesecake chili chutney cornbread crepe curry gingerbread meatloaf pancake pasta pizza potato salad punch quiche salsa	<table border="1"> <thead> <tr> <th colspan="6">Representative examples of categories</th> </tr> <tr> <th>Category 1</th> <th>Category 2</th> <th>Category 3</th> <th>Category 4</th> <th>Category 5</th> <th>Category 6</th> </tr> <tr> <th>Example</th> <th>Example</th> <th>Example</th> <th>Example</th> <th>Example</th> <th>Example</th> </tr> </thead> <tbody> <tr> <td>chili powder</td> <td>brown sugar</td> <td>brandy</td> <td>zest</td> <td>powder sugar</td> <td>vinegar</td> </tr> <tr> <td>broth</td> <td>soda</td> <td>soda</td> <td>jam or jelly</td> <td>cracker</td> <td>apple</td> </tr> <tr> <td>cumin</td> <td>egg</td> <td>ice</td> <td>apricot</td> <td>sour cream</td> <td>cinnamon</td> </tr> <tr> <td>garlic</td> <td>flour</td> <td>pine nut</td> <td>buttermilk</td> <td>cream cheese</td> <td>cranberry</td> </tr> <tr> <td>onion</td> <td>ginger</td> <td>sugar</td> <td>powder sugar</td> <td>egg</td> <td>ginger</td> </tr> <tr> <td>pepper</td> <td>milk</td> <td>watermelon</td> <td>whipped cream</td> <td>pie filling</td> <td>raisin</td> </tr> <tr> <td>tomato juice</td> <td>butter</td> <td>water</td> <td>cream cheese</td> <td>sugar</td> <td>sugar</td> </tr> <tr> <td>tomato paste</td> <td>salt</td> <td></td> <td>egg</td> <td>vanilla</td> <td></td> </tr> <tr> <td>oil</td> <td></td> <td></td> <td>flour</td> <td>butter</td> <td></td> </tr> <tr> <td>turkey</td> <td></td> <td></td> <td>honey</td> <td></td> <td></td> </tr> <tr> <td>black bean</td> <td></td> <td></td> <td>vanilla</td> <td></td> <td></td> </tr> <tr> <td>corn</td> <td></td> <td></td> <td>milk</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td>salt</td> <td></td> <td></td> </tr> </tbody> </table>	Representative examples of categories						Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Example	Example	Example	Example	Example	Example	chili powder	brown sugar	brandy	zest	powder sugar	vinegar	broth	soda	soda	jam or jelly	cracker	apple	cumin	egg	ice	apricot	sour cream	cinnamon	garlic	flour	pine nut	buttermilk	cream cheese	cranberry	onion	ginger	sugar	powder sugar	egg	ginger	pepper	milk	watermelon	whipped cream	pie filling	raisin	tomato juice	butter	water	cream cheese	sugar	sugar	tomato paste	salt		egg	vanilla		oil			flour	butter		turkey			honey			black bean			vanilla			corn			milk						salt		
Representative examples of categories																																																																																																		
Category 1	Category 2	Category 3	Category 4	Category 5	Category 6																																																																																													
Example	Example	Example	Example	Example	Example																																																																																													
chili powder	brown sugar	brandy	zest	powder sugar	vinegar																																																																																													
broth	soda	soda	jam or jelly	cracker	apple																																																																																													
cumin	egg	ice	apricot	sour cream	cinnamon																																																																																													
garlic	flour	pine nut	buttermilk	cream cheese	cranberry																																																																																													
onion	ginger	sugar	powder sugar	egg	ginger																																																																																													
pepper	milk	watermelon	whipped cream	pie filling	raisin																																																																																													
tomato juice	butter	water	cream cheese	sugar	sugar																																																																																													
tomato paste	salt		egg	vanilla																																																																																														
oil			flour	butter																																																																																														
turkey			honey																																																																																															
black bean			vanilla																																																																																															
corn			milk																																																																																															
			salt																																																																																															

Figure 4: Web-interface for human subject experiment

this task, as it required clusters to be well-explained for subjects to perform classifications, but did not require special expertise or training.

Our experiment used a within-subjects design, allowing for more powerful statistical testing and to mitigate the effects of inter-participant variability. To account for possible learning effects, we blocked the LCM and LDA questions and balanced the assignment of participants into the two ordering groups. Half the subjects saw all 8 LCM questions first, the other half saw the 8 LDA questions first. Twenty-four participants (10 female, 14 males, average age 27) performed the task, answering a total of 384 questions. Subjects were encouraged to answer the questions as quickly and accurately as possible, and were forced to take five second break in every four questions to mitigate fatigue effects.

Cluster representations (i.e. explanations) from LDA were presented as the set of top ingredients for each recipe topic cluster. LCM simply presented the prototype of each cluster. For fairness, the number of top ingredients presented from LDA was set as the number of ingredients from the corresponding LCM prototype. Also, LCM subspaces were not presented to participants in this experiment. We ran Gibbs sampling for LDA with different initializations until the ground truth clusters were visually identifiable.

Using explanations from LCM, the average accuracy of classification is 85.9%, which was statistically significantly higher ($c^2(1, N = 24) = 12.15, p \ll 0.001$) than that of LDA, 71.3%. For both LDA and LCM, the each ground truth label was manually coded by two domain experts, the first author and one independent analyst (kappa coefficient: 1). Note that these manually produced ground truth labels were identical to the labels that LDA and LCM predicted for each recipe. There was no statistically significant different in time spent on each questions with LDA and LCM ($t(24) = 0.89, p = 0.37$) — the overall average was 32 seconds per question, with 3% more time on LCM than on LDA. Subjective evaluation using Likert-style questionnaires produced no statistically significant differences in reported preferences for LDA versus LCM.

The experiment demonstrated that prototypes better supported participants in correctly performing the classification task. Interesting, participants appeared to not have insight into their superior performance using output of LCM versus LDA.

References

- [1] Agnar Aamodt. A knowledge-intensive, integrated approach to problem solving and sustained learning. *Knowledge Engineering and Image Processing Group. University of Trondheim*, pages 27–85, 1991.
- [2] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.
- [3] Cedric Archambeau, Balaji Lakshminarayanan, and Guillaume Bouchard. Latent IBP compound dirichlet allocation.
- [4] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 99:1803–1831, 2010.
- [5] Isabelle Bichindaritz and Cindy Marling. Case-based reasoning in the health sciences: What’s next? *Artificial intelligence in medicine*, 36(2):127–135, 2006.
- [6] Jacob Bien, Robert Tibshirani, et al. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011.
- [7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022, 2003.
- [8] John S Carroll. Analyzing decision behavior: The magician’s audience. *Cognitive processes in choice and decision behavior*, pages 69–76, 1980.
- [9] Jonathan Chang, Jordan L Boyd-Graber, Sean Gerrish, Chong Wang, and David M Blei. Reading tea leaves: How humans interpret topic models. In *NIPS*, volume 22, pages 288–296, 2009.
- [10] Marvin S Cohen, Jared T Freeman, and Steve Wolf. Metarecognition in time-stressed decision making: Recognizing, critiquing, and correcting. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 38(2):206–219, 1996.
- [11] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.
- [12] Pádraig Cunningham, Dónal Doyle, and John Loughrey. An evaluation of the usefulness of case-based explanation. In *Case-Based Reasoning Research and Development*, pages 122–130. Springer, 2003.
- [13] Glenn De’ath and Katharina E Fabricius. Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, 81(11):3178–3192, 2000.
- [14] Jacob Eisenstein, Amr Ahmed, and Eric P Xing. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1041–1048, 2011.
- [15] John T Favata and Geetha Srikantan. A multiple feature/resolution approach to handprinted digit and character recognition. *International journal of imaging systems and technology*, 7(4):304–311, 1996.
- [16] Alex A Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, 2014.
- [17] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.
- [18] N Hagita, S Naito, and I Masuda. Handprinted kanji characters recognition based on pattern matching method. *ICTP*, pages 169–174, 1983.
- [19] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.
- [20] Jonathan J. Hull. A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(5):550–554, 1994.
- [21] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

- [22] Gwang-Hee Kim, Sung-Hoon An, and Kyung-In Kang. Comparison of construction cost estimating models based on regression analysis, neural networks, and case-based reasoning. *Building and environment*, 39(10):1235–1242, 2004.
- [23] Gary A Klein. Do decision biases explain too much. *Human Factors Society Bulletin*, 32(5):1–3, 1989.
- [24] Janet L Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34, 1992.
- [25] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.
- [26] Hui Li and Jie Sun. Ranking-order case-based reasoning for financial distress prediction. *Knowledge-Based Systems*, 21(8):868–878, 2008.
- [27] Raanan Lipshitz. Converging themes in the study of decision making in realistic settings. 1993.
- [28] CL Liu, YJ Liu, and RW Dai. Preprocessing and statistical/structural feature extraction for handwritten numeral recognition. *Progress of Handwriting Recognition, World Scientific, Singapore*, pages 161–168, 1997.
- [29] James G March. Theories of choice and making decisions. *Society*, 20(1):29–39, 1982.
- [30] J William Murdock, David W Aha, and Leonard A Breslow. Assessing elaborated hypotheses: An interpretive case-based reasoning approach. In *Case-Based Reasoning Research and Development*, pages 332–346. Springer, 2003.
- [31] Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, 1972.
- [32] Nancy Pennington and Reid Hastie. A theory of explanation-based decision making. 1993.
- [33] Xuan-Hieu Phan and Cam-Tu Nguyen. Gibbslda++, ac/c++ implementation of latent dirichlet allocation (lda) using gibbs sampling for parameter estimation and inference, 2013.
- [34] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.
- [35] R.C. Schank. *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge University Press, 1982.
- [36] Stephen Slade. Case-based reasoning: A research paradigm. *AI magazine*, 12(1):42, 1991.
- [37] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in case-based reasoning—perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143, 2005.
- [38] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- [39] Sinead Williamson, Chong Wang, Katherine Heller, and David Blei. The IBP compound dirichlet process and its application to focused topic modeling. 2010.
- [40] Jun Zhu, Amr Ahmed, and Eric P Xing. Medlda: maximum margin supervised topic models. *The Journal of Machine Learning Research*, 13(1):2237–2278, 2012.

