7.5.3 Implementing Categories Defined by Probability and Similarity

Many categories cannot be defined in terms of required properties, and instead must be defined probabilistically, where category membership is determined by properties that resources are likely to share. Consider the category "friend." You probably consider many people to be your friends, but you have longtime friends, school friends, workplace friends, friends you see only at the gym, and friends of your parents. Each of these types of friends represents a different cluster of common properties. If someone is described to you as a potential friend or date, how accurately can you predict that the person will become a friend? (See the sidebar, Finding Friends and Dates: Lessons for Learning Categories (page 393))

Probabilistic categories can be challenging to define and use because it can be difficult to keep in mind the complex feature correlations and probabilities exhibited by different clusters of instances from some domain. Furthermore, when the category being learned is broad with a large number of members, the sample from which you learn strongly shapes what you learn. For example, people who grow up in high-density and diverse urban areas may have less predictable ideas of what an acceptable potential date looks like than someone in a remote rural area with a more homogeneous population.

More generally, if you are organizing a domain where the resources are active, change their state, or are measurements of properties that vary and co-occur probabilistically, the sample you choose strongly affects the accuracy of models for classification or prediction. In *The Signal and the Noise,* statistician Nate Silver explains how many notable predictions failed because of poor sampling techniques. One common sampling mistake is to use too short a historical window to assemble the training dataset; this is often a corollary of a second mistake, an over reliance on recent data because it is more available. For example, the collapse of housing prices and the resulting financial crisis of 2008 can be explained in part because the models that lenders used to predict mortgage foreclosures were based on data from 1980-2005, when house prices tended to grow higher. As a result, when mortgage foreclosures increased rapidly, the results were "out of sample" and were initially misinterpreted, delaying responses to the crisis.

Samples from dynamic and probabilistic domains result in models that capture this variability. Unfortunately, because many forecasters want to seem authoritative, and many people do not understand probability, classifications or predictions that are inherently imprecise are often presented with certainty and exactness even though they are probabilistic with a range of outcomes. Silver tells the story of a disastrous 1997 flood caused when the Red River crested at 54 feet when the levees protecting the town of Grand Forks were at 51 feet. The

Finding Friends and Dates: Lessons for Learning Categories

Online dating or matchmaking sites use many of the same features to describe people, but also have additional features to make more accurate matches for their targeted users. As the number of features grows, there are exponentially more combinations of shared properties. For example, the matchmaking site eHarmony employs 29 “Dimensions of Compatibility” and more than 200 questions to create a user profile. Even if the 29 dimensions were Boolean (would you describe yourself as x?) this yields 2^{29} or over 500,000,000 different combinations. Using these complex resource descriptions to predict the probability of a good match requires matchmaking sites to use proprietary machine learning algorithms to propose matches, which are ranked with unexplained measures and precision (what does an 80% match mean?). Not surprisingly, many people who try online dating give up after less success than they expected.

With such a large number of features in user profiles, any matching algorithm confronts what machine learning calls the curse of dimensionality. With high-dimensional data, there can never be enough instances to learn which features are really the most important. Neither you nor the online dating algorithm will ever meet enough different kinds of people to reliably predict the outcome of a possible match.

But all is not hopeless. Machine learning programs attack the curse of dimensionality using statistical techniques that use correlations among features to combine them or adjust the weights given to features to reflect their value in making predictions or classifications. For example, OKCupid asks people to rate how much importance they assign to match questions. You might prefer cats to dogs, and you might either never consider dating a dog lover or you might not care at all.

Another way to reduce the number of features needed to classify accurately is to reduce the scope of the category being learned. The matchmaking model for sites that target people with particular professions, religions, or political views would be less complex than the eHarmony one, because the former will have fewer relevant features, and hence fewer random correlations and noise that will undermine its accuracy. All other things being equal, the lower the variability in a set of examples, the better a model that learns from that data will perform.

weather service had predicted a crest between 40 and 58 feet, but emphasized the midpoint of the range, which was 49 feet. Unfortunately, most people interpreted this probabilistic prediction as if it were a binary classification, “flood” versus “no flood,” ignored the range of the forecast, and failed to prepare for a flood that had about a 35% chance of occurring.[444][DS]

7.5.3.1 Probabilistic Decision Trees

In §7.5.2, we showed how a rule-based decision tree could be used to implement a strict property-based classification in which a bank uses tests for the properties of “annual income” and “monthly loan payment” to classify applicants as approved or denied. We can adapt that example to illustrate probabilistic decision trees, which are better suited for implementing categories in which category membership is probabilistic rather than absolute.

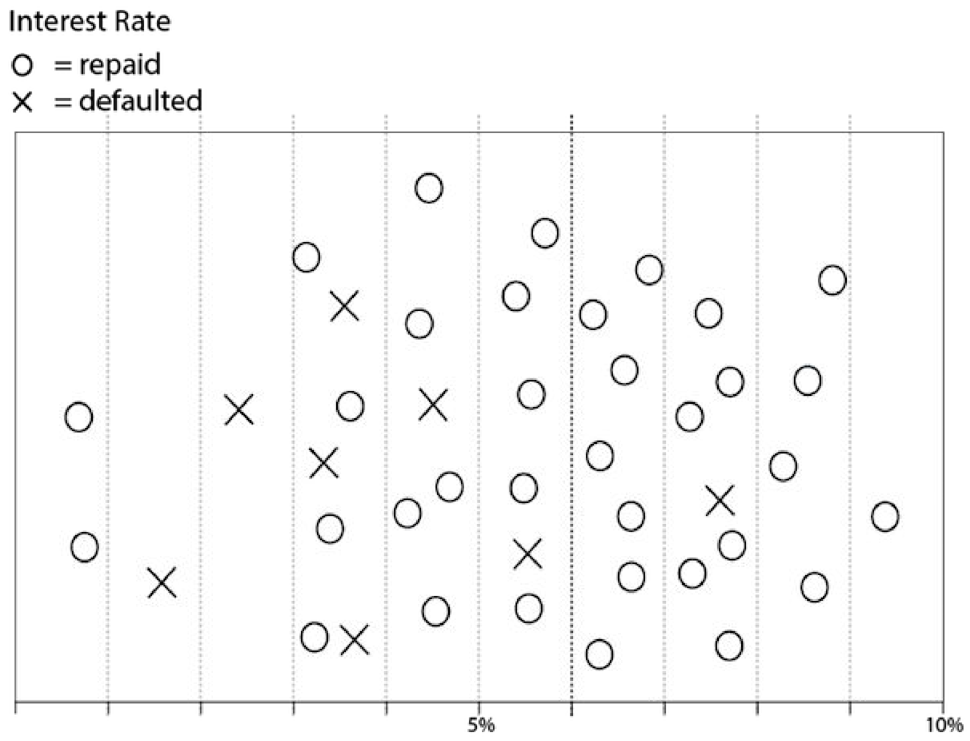
Banks that are more flexible about making loans can be more profitable because they can make loans to people that a stricter bank would reject but who still are able to make loan payments. Instead of enforcing conservative and fixed cutoffs on income and monthly payments, these banks consider more properties and look at applications in a more probabilistic way. These banks recognize that not every loan applicant who is likely to repay the loan looks exactly the same; “annual income” and “monthly loan payment” remain important properties, but other factors might also be useful predictors, and there is more than one configuration of values that an applicant could satisfy to be approved for a loan.

Which properties of applicants best predict whether they will repay the loan or default? A property that predicts each at 50% isn’t helpful because the bank might as well flip a coin, but a property that splits the applicants into two sets, each with very different probabilities for repayment and defaulting, is very helpful in making a loan decision.

A data-driven bank relies upon historical data about loan repayment and defaults to train algorithms that create decision trees by repeatedly splitting the applicants into subsets that are most different in their predictions. Subsets of applicants with a high probability of repayment would be approved, and those with a high probability of default would be denied a loan. One method for selecting the property test for making each split is calculating the “information gain” (see the sidebar Using “Information Theory” to Quantify Organization (page 78)). This measure captures the degree to which each subset contains a “pure” group in which every applicant is classified the same, as likely repayers or likely defaulters.

For example, consider the chart in Figure 7.2, Historical Data: Loan Repayment Based on Interest Rate which is a simplified representation of the bank’s historical data on loan defaults based on the initial interest rate. The chart represents loans that were repaid with “o” and those that defaulted with “x.” Is there an interest rate that divides them into “pure” sets, one that contains only “o” loans and the other that contains only “x” loans?

Figure 7.2. Historical Data: Loan Repayment Based on Interest Rate

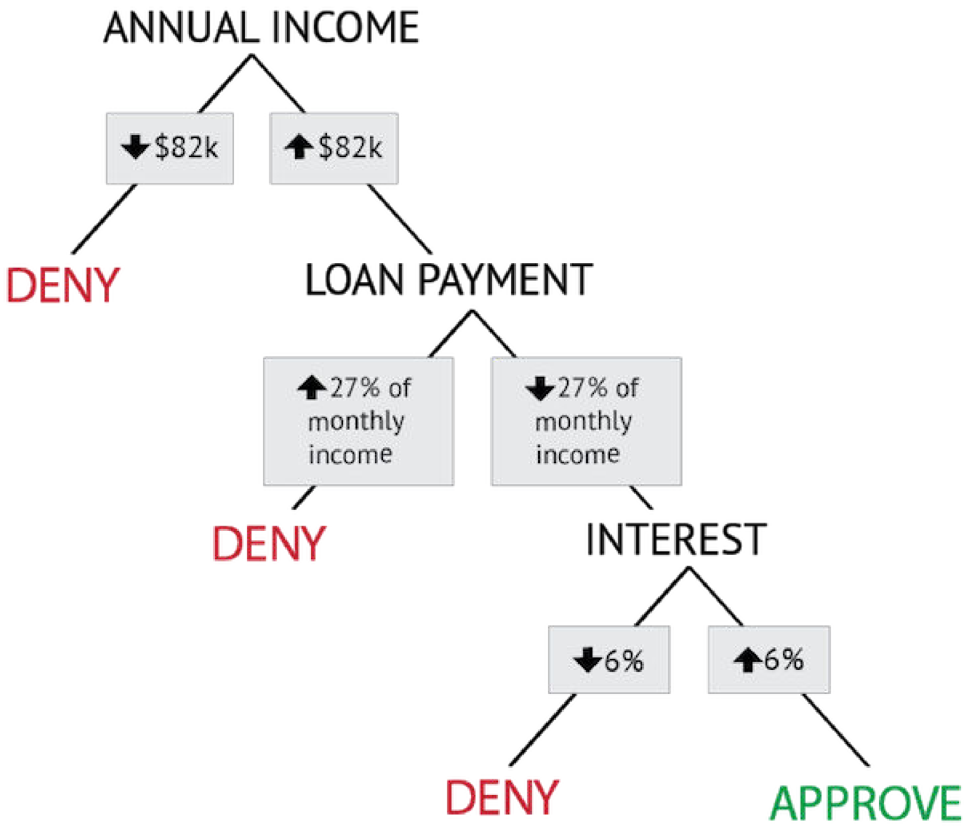


The "o" symbol represents loans that were repaid by the borrower; "x" represents loans on which the borrower defaulted. A 6% rate (darker vertical line) best divides the loans into subsets that differ in the payment outcome.

You can see that no interest rate divides these into pure sets. So the best that can be done is to find the interest rate that divides them so that the proportions of defaulters are most different on each side of the line.445[DS]

This dividing line at the 6% interest rate best divides those who defaulted from those who repaid their loan. Most people who borrowed at 6% or greater repaid the loan, while those who took out loans at a lower rate were more likely to default. This might seem counter-intuitive until you learn that the lower-interest rate loans had adjustable rates that increased after a few years, causing the monthly payments to increase substantially. More prudent borrowers were willing to pay higher interest rates that were fixed rather than adjustable to avoid radical increases in their monthly payments.

Figure 7.3. Probabilistic Decision Tree



In this probabilistic decision tree, the sequence of property tests and the threshold values in each test divide the loan applicants into categories that differ in how likely they are to repay the loan.

This calculation is carried out for each of the attributes in the historical data set to identify the one that best divides the applicants into the repaid and defaulted categories. The attributes and the value that defines the decision rule can then be ordered to create a decision tree similar to the rule-based one we saw in §7.5.2. In our hypothetical case, it turns out that the best order in which to test the properties is Income, Monthly Payment, and Interest Rate, as shown in Figure 7.3, Probabilistic Decision Tree. The end result is still a set of rules, but behind each decision in the tree are probabilities based on historical data that can more accurately predict whether an applicant will repay or default. Thus, instead of the arbitrary cutoffs at $100,000 in income and 25% for monthly payment, the bank can offer loans to people with lower incomes and remain profitable doing so, because it knows from historical data that $82,000 and 27% are the optimal decision points. Using the interest rate in their decision process is an additional test to ensure that people can afford to make loan payments even if interest rates go up.[446][Bus]

Because decision trees specify a sequence of rules that make property tests, they are highly interpretable, which makes them a very popular choice for data scientists building models much more complex than the simple loan example here. But they assume that every class is a conjunction of all the properties used to define them. This makes them susceptible to over-fitting because if they grow very deep with many property conjunctions, they capture exactly the properties that describe each member of the training set, effectively memorizing the training data. In other words, they capture both what is generally true beyond the set and what is particular to the training set only, when the goal is to build a model that captures only what is generally true. Overfitting in decision trees can be prevented by pruning back the tree after it has perfectly classified the training set, or by limiting the depth of the tree in advance, essentially pre-pruning it.

7.5.3.2 Naïve Bayes Classifiers

Another commonly used approach to implement a classifier for probabilistic categories is called Naïve Bayes. It employs Bayes' Theorem for learning the importance of a particular property for correct classification. There are some common sense ideas that are embodied in Bayes' Theorem:

- When you have a hypothesis or prior belief about the relationship between a property and a classification, new evidence consistent with that belief should increase your confidence.
- Contradictory evidence should reduce confidence in your belief.
- If the base rate for some kind of event is low, do not forget that when you make a prediction or classification for a new specific instance. It is easy to be overly influenced by recent information.

Now we can translate these ideas into calculations about how learning takes place. For property A and classification B, Bayes' Theorem says:

$$P(A \mid B) = P(B|A)\, P(A) / P(B)$$

The left hand side of the equation, P (A | B), is what we want to estimate but can't measure directly: the probability that A is the correct classification for an item or observation that has property B. This is called the conditional or posterior probability because it is estimated after seeing the evidence of property B.

P (B | A) is the probability that any item correctly classified as A has property B. This is called the likelihood function.

P (A) and P (B) are the independent or prior probabilities of A and B; what proportion of the items are classified as A? How often does property B occur in some set of items?

Using Bayes' Theorem to Calculate Conditional Probability

Your personal library contains 60% fiction and 40% nonfiction books. All of the fiction books are in ebook format, and half of the nonfiction books are ebooks and half are in print format. If you pick a book at random and it is in ebook format, what is the probability that it is nonfiction?

Bayes' Theorem tells us that:

$$P\ (\text{nonfiction} \mid \text{ebook}) = P\ (\text{ebook} \mid \text{nonfiction}) \times P\ (\text{nonfiction}) / P\ (\text{ebook}).$$

We know: $P\ (\text{ebook} \mid \text{nonfiction}) = .5$ and $P\ (\text{nonfiction}) = .4$

We compute P (ebook) using the law of total probability to compute the combined probability of all the independent ways in which an ebook might be sampled. In this example there are two ways:

$$\begin{aligned} P\ (\text{ebook}) &= P\ (\text{ebook} \mid \text{nonfiction}) \times P\ (\text{nonfiction}) \\ &\quad + P\ (\text{ebook} \mid \text{fiction}) \times P\ (\text{fiction}) \\ &= (.5 \times .4) + (1 \times .6) = .8 \end{aligned}$$

Therefore: $P\ (\text{nonfiction} \mid \text{ebook}) = (.5 \times .4) / .8 = .25$

Now let's apply Bayes' Theorem to implement email spam filtering. Messages are classified as SPAM or HAM (i.e., non-SPAM); the former are sent to a SPAM folder, while the latter head to your inbox.

1. Select Properties. We start with a set of properties, some from the message metadata like the sender's email address or the number of recipients, and some from the message content. Every word that appears in messages can be treated as a separate property[447][Com]
2. Assemble Training Data. We assemble a set of email message that have been correctly assigned to the SPAM and HAM categories. These labeled instances make up the training set.
3. Analyze the Training Data. For each message, does it contain a particular property? For each message, is it classified as SPAM? If a message is classified as SPAM, does it contain a particular property? (These are the three probabilities on the right side of the Bayes equation).
4. Learn. The conditional probability (the left side of the Bayes equation) is recalculated, adjusting the predictive value of each property. Taken together, all of the properties are now able to correctly assign (most of) the messages into the categories they belonged to in the training set.

5. Classify. The trained classifier is now ready to classify uncategorized messages to the SPAM or HAM categories.
6. Improve. The classifier can improve its accuracy if the user gives it feedback by reclassifying SPAM messages as HAM ones or vice versa. The most efficient learning occurs when an algorithm uses “active learning” techniques to choose its own training data by soliciting user feedback only where it is uncertain about how to classify a message. For example, the algorithm might be confident that a message with “Cheap drugs” in the subject line is SPAM, but if the message comes from a longtime correspondent, the algorithm might ask the user to confirm that the classification.[448][Com]

7.5.3.3 Categories Created by Clustering

In the previous two sections we discussed how probabilistic decision trees and naïve Bayes classifiers implement categories that are defined by typically shared properties and similarity. Both are examples of supervised learning because they need correctly classified examples as training data, and they learn the categories they are taught.

In contrast, clustering techniques are unsupervised; they analyze a collection of uncategorized resources to discover statistical regularities or structure among the items, creating a set of categories without any labeled training data.

Clustering techniques share the goal of creating meaningful categories from a collection of items whose properties are hard to directly perceive and evaluate, which implies that category membership cannot easily be reduced to specific property tests and instead must be based on similarity. For example, with large sets of documents or behavioral data, clustering techniques can find categories of documents with the same topics, genre, or sentiment, or categories of people with similar habits and preferences.

Because clustering techniques are unsupervised, they create categories based on calculations of similarity between resources, maximizing the similarity of resources within a category and maximizing the differences between them. These statistically-learned categories are not always meaningful ones that can be named and used by people, and the choice of properties and methods for calculating similarity can result in very different numbers and types of categories. Some clustering techniques for text resources suggest names for the clusters based on the important words in documents at the center of each cluster. However, unless there is a labeled set of resources from the same domain that can be used as a check to see if the clustering discovered the same categories, it is up to the data analyst or information scientist to make sense of the discovered clusters or topics.

There are many different distance-based clustering techniques, but they share three basic methods.

- The first shared method is that clustering techniques start with an initially uncategorized set of items or documents that are represented in ways that enable measures of inter-item similarity can be calculated. This representation is most often a vector of property values or the probabilities of different properties, so that items can be represented in a multidimensional space and similarity calculated using a distance function like those described in §7.3.6.2 Geometric Models of Similarity (page 378).[449][Com]
- The second shared method is that categories are created by putting items that are most similar into the same category. Hierarchical clustering approaches start with every item in its own category. Other approaches, notably one called "K-means clustering," start with a fixed number of K categories initialized with a randomly chosen item or document from the complete set.
- The third shared method is refining the system of categories by iterative similarity recalculation each time an item is added to a category. Approaches that start with every item in its own category create a hierarchical system of categories by merging the two most similar categories, recomputing the similarity between the new category and the remaining ones, and repeating this process until all the categories are merged into a single category at the root of a category tree. Techniques that start with a fixed number of categories do not create new ones but instead repeatedly recalculate the "centroid" of the category by adjusting its property representation to the average of all its members after a new member is added.[450][Com]

It makes sense that the algorithms that create clusters or categories of similar items can be later used as classifiers by using the same similarity measures to compare the unclassified items against items that are labeled by category. There are different choices about which items to compare with the unclassified one:

- The centroid: a prototypical or average item calculated on the properties of all the category members. However, the centroid might not correspond to any actual member (see the sidebar Median versus Average (page 121)), and this can make it hard to interpret the classification.
- Items that actually exist: Because the items in categories defined by similarity are not equally typical or good members, it is more robust to test against more than one exemplar. Classifiers that use this approach are called nearest-neighbor techniques, and they essentially vote among themselves and the majority category is assigned to the new item.

- The edge cases: These are instances that are closest to the boundary between two categories, so there need to be at least two of them, one in each category. Because they are not typical members of the category, they are the hardest to classify initially, but using them in classifiers emphasizes the properties that are the most discriminating. This is the approach taken by support vector machines, which are not clustering algorithms but are somewhat like nearest-neighbor algorithms in that they calculate the similarity of an unclassified item to these edge cases. Their name makes more sense if you think of the vectors that represent the "edge cases" being used to "support" the category boundary, which falls between them.

7.5.3.4 Neural networks

Among the best performing classifiers for categorizing by similarity and probabilistic membership are those implemented using neural networks, and especially those employing deep learning techniques. Deep learning algorithms can learn categories from labeled training data or by using autoencoding, an unsupervised learning technique that trains a neural network to reconstruct its input data. However, instead of using the properties that are defined in the data, deep learning algorithms devise a very large number of features in hidden hierarchical layers, which makes them uninterpretable by people. The key idea that made deep learning possible is the use of "backpropagation" to adjust the weights on features by working backwards from the output (the object classification produced by the network) all the way back to the input. The use of deep learning to classify images was mentioned in §5.4.2.[451][DS]

7.5.4 Implementing Goal-Based Categories

Goal-based categories are highly individualized, and are often used just once in a very specific context. However, it is useful to consider that we could implement model goal-derived categories as rule-based decision trees by ordering the decisions to ensure that any sub-goals are satisfied according to their priority. We could understand the category "Things to take from a burning house" by first asking the question "Are there living things in the house?" because that might be the most important sub-goal. If the answer to that question is "yes," we might proceed along a different path than if the answer is "no." Similarly, we might put a higher priority on things that cannot be replaced (Grandma's photos) than those that can (passport).

7.5.5 Implementing Theory-Based Categories

Theory-based categories arise in domains in which the items to be categorized are characterized by abstract or complex relationships with their features and with each other. With this model an entity need not be understood as inherently possessing features shared in common with another entity. Rather, people project features from one thing to another in a search for congruities between things, much as clue receivers in the second round of the *Pyramid* game search for congruities between examples provided by the clue giver in order to guess the target category. For example, a clue like “screaming baby” can suggest many categories, as can “parking meter.” But the likely intersection of the interactions one can have with babies and parking meters is that they are both “Things you need to feed.”

Theory-based categories are created as cognitive constructs when we use analogies and classify, because things brought together by analogy have abstract rather than literal similarity. The most influential model of analogical processing is Structure Mapping, whose development and application has been guided by Dedre Gentner for over three decades.

The key insight in Structure Mapping is that an analogy “a T is like B” is created by matching relational structures and not properties between the base domain B and a target domain T. We take any two things, analyze the relational structures they contain, and align them to find correspondences between them. The properties of objects in the two domains need not match, and in fact, if too many properties match, analogy goes away and we have literal similarity:

- Analogy: The hydrogen atom is like our solar system
- Literal Similarity: The X12 star system in the Andromeda galaxy is like our solar system

Structure Mapping theory was implemented in the Structure-Mapping Engine (SME), which both formalized the theory and offered a computationally-tractable algorithm for carrying out the process of mapping structures and drawing inferences.[452][CogSci]