Model-Based Classification

David M. Blei

February 29, 2012

Probability models

- A probability model is a joint distribution of a set of observations.

- Often, a model is indexed by a *parameter*. Each value of the parameter gives a different distribution of the data.
  - The parameter of a Bernoulli is the probability of heads.
  - The parameters of a Gaussian are its mean and variance.

- Many models (but not all) assume the data are *independent and identically distributed*.

- For a boring example, consider N coin flips, each of which has heads with probability π,

$$p(x_1, \ldots, x_N \mid \pi) = \prod_{n=1}^{N} p(x_i \mid \pi). \tag{1}$$

  Each term is a Bernoulli,

$$p(x_n \mid \pi) = \pi^{1(x_n = \mathrm{h})}(1-\pi)^{1(x_n = \mathrm{t})} \tag{2}$$

- Suppose we flip a coin N times and record the outcomes.

- Further suppose that *we think* that the probability of heads is π. (This is distinct from whatever the probability of heads "really" is.)

- Given π, the probability of an observed sequence is

$$p(x_1, \ldots, x_N \mid \pi) = \prod_{n=1}^{N} \pi^{1[x_n = \mathrm{h}]}(1-\pi)^{1[x_n = \mathrm{t}]} \tag{3}$$

- As a function of π, the probability of a data set is the *likelihood function*.

- Taking logs, this is the *log likelihood function*.

$$\mathscr{L}(\pi) = \sum_{n=1}^{N} 1[x_n = \mathrm{h}] \log \pi + 1[x_n = \mathrm{t}] \log(1 - \pi) \tag{4}$$

- The *maximum likelihood estimate* is the value of the parameter that maximizes the log likelihood (equivalently, the likelihood).

- In the Bernoulli example, it is the proportion of heads.

$$\hat{\pi} = \frac{1}{N} \sum_{n=1}^{N} 1[x_n = H] \tag{5}$$

- In a Gaussian, it is the empirical mean

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{6}$$

  and empirical variance

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{\mu})^2 \tag{7}$$

- In a sense, this is the value that best explains our observations.

- Why is the MLE good?
  - The MLE is *consistent*.
  - Flip a coin N times with true bias π^*.
  - Estimate the parameter from $x_1, \ldots x_N$ with the MLE $\hat{\pi}$.
  - Then,

$$\lim_{N \to \infty} \hat{\pi} = \pi^*$$

  - This is a good thing. It lets many statisticians sleep at night.

- (R demonstration: Bernoulli data sets)

- (Slides: Modeling approval ratings with a Gaussian)

Graphical models

- Represents a joint distribution of random variables; used to show models.
  (Also called "Bayesian network")

- Semantics:
  - Nodes are RVs; Edges denote possible dependence
  - Shaded nodes are observed; unshaded nodes are hidden
  - GMs with shaded nodes represent posterior distributions.

- *Each graphical model is a family of distributions.*

- Connects computing about models to graph structure (COS513)

- Connects independence assumptions to graph structure (COS513)

- Here we'll use it as a schematic for factored joint distributions.
  (Show the classification graphical model.)

- Return briefly to the Aliens/Watch example
  - It's true that many members of this family do not have $X \perp\!\!\!\perp Y \mid Z$.
  - The class discussion revealed conditions where Z is dependent on X and Y, but they are still conditionally independent. (That is a subfamily of this family.)
  - I conjectured that for conditional independence between X and Y, one of the dependencies of Z had to be broken. However, I couldn't prove it.

Classification set-up

- In classification we observe two sets of data. One set contains data ("features") labeled with a category. The other set contains unlabeled features.

- The idea is to fit a model of how the label relates to the features. Then given new unlabeled data, predict the category. This is *supervised learning*.

- More formally,

  - The fully observed data (also called the "training data") are $\{x_i, c_i\}_{i=1}^n$, where c_i is in one of k categories and the features x_i are a vector of values.
  - The partially observed data are x_{new}. Our goal is to predict y_{new}.

- Here are some examples. In each, what are the features? What are the labels?
  - Classify images into semantic categories
  - Classify news articles into section of the newspaper
  - Classify genetic code as entron or intron
  - Classify radar blips as friendly or unfriendly
  - Classify credit cards as stolen or not stolen
  - Others?

Basic idea

- Classification with generative models links statistical modeling to classification problems.

- We can model many kinds of data (features x) with appropriate probability distributions.
  - Continuous features can be modeled with a Gaussian
  - Binary features can be modeled with a Bernoulli
  - Positive features can be modeled with a Gamma
  - Etc.

- Recall that our training set is a collection of labeled feature vectors (x_i, c_i). The idea is to fit a distribution of features *conditional* on the class label.
  - A different Gaussian for each class label
  - A different Bernoulli for each class label
  - A different Gamma distribution for each class label
  - Etc.

- To classify a new feature vector x we compute the conditional distribution of the class label given the features $p(c \mid x)$.

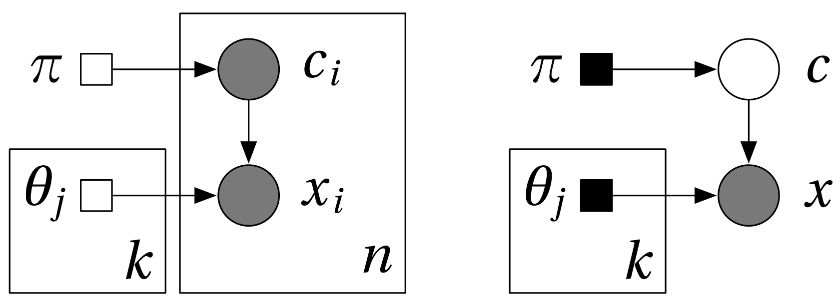
- For example:

- Suppose images are represented with continuous value image features such as color intensities, texture features, and others.
- Each training image is labeled with one category, such as “outdoor”, “indoor”, “sports”. (This data set exists. It’s called CalTech 101.)
- To build our classifier, we fit a Gaussian distribution to each feature conditional on the class label. For example, we would find a distribution of color for indoor scenes, for outdoor scenes, for portraits, and each other category.[1]
- To classify a new image, we would consider each label and look at the probability of its features given that label. (It’s a little more complicated than this—see below—and this process will emerge naturally from Bayes rule.)

Modeling assumptions

- Data
  - The training data are $\mathscr{D} = \{x_i, c_i\}_{i=1}^n$
  - The data to predict are $\{x_{\text{new}}\}$.

- The graphical models for fitting and prediction are the following,



- Some details
  - Small boxes are parameters. Unshaded are fit; shaded are fixed.
  - The joint distribution of a feature vector and label is

$$p(x, c \mid \pi, \theta_{1:k}) = p(c \mid \pi) p(x \mid c, \theta_{1:k}) \tag{8}$$

  - The parameters $\theta_{1:k}$ are the conditional distributions of the features.
    The parameter π is the probability of seeing each class.

[1] Actually, we could fit a conditional distribution for the whole vector using a multivariate Gaussian. But, for now, let’s assume we fit one distribution for each feature.

- The second term $p(x \mid c, \theta_{1:k})$ selects the right class. (More below.)

- How do we "select" the right class in $p(x \mid c, \theta_{1:k})$ and $p(c \mid \pi)$?
  - The class label c is represented as a k-vector with a single one.
    For example, a data point in the fourth class has $c = \langle 0, 0, 0, 1, 0, 0, 0 \rangle$.
  - The second term is
    $$p(x \mid c, \theta_{1:k}) = \prod_{j=1}^{k} p(x \mid \theta_j)^{c^j}. \quad (9)$$
    (Confirm that this equals the intended probability.)
  - The distribution in π works the same way. The parameter π is a k-vector of probabilities that sum to one. The probability of selecting a particular class label is
    $$p(c \mid \pi) = \prod_{j=1}^{k} \pi_j^{c^j}. \quad (10)$$
    Note: the space of positive vectors that sum to one is called the *simplex*.
  - As we'll see, this representation helps in fitting the model.

- In *fitting*, we want to find the class parameters θ_k and class proportions π from a data set of labeled observations.

- In *prediction*, we use fitted parameters to predict the label of an unlabeled data point. We need to compute $p(c \mid x)$.

- Example #1 : Gaussian classification
  - Let's say the data are continuous.
  - To be concrete, suppose each x_i is an image, with a vector real-valued image features measured on it. (E.g., these might be texture, color histogram, etc.)
  - Suppose each class is described by a Gaussian, where $\theta_k = \mu_k$ and $x \mid \theta_k \sim \mathcal{N}(\mu_k, \sigma^2)$. (Each shares the same variance, for simplicity.)
  - In fitting, we'd find the Gaussian distributions that best describe each class.
  - In prediction, we can take new images and classify them.

- Example #2 : Multinomial classification
  - Let's say the data are discrete.

- To be concrete, suppose each x_i is a document, i.e., a collection of observed words from a vocabulary.
- Suppose each class is described by a multinomial distribution. In this case, θ_k is a distribution over terms and we assume the words of each document are drawn independently from that distribution. That is,

$$p(x \mid \theta_k) = \prod_{j=1}^{\ell} \theta_{k,x_j}, \tag{11}$$

where x_j is the jth word in document x (of length ℓ).
- Actually, we'll use the "selection" mechanism here too. This lets us see how the probability is only a function of the count of each word. (For this reason, models like this are often called "bag of words" models.)
- First, write down the previous equation in this form

$$p(x \mid \theta_k) = \prod_{j=1}^{\ell} \prod_{v=1}^{V} \theta_{kv}^{x_j^v} \tag{12}$$

- How many times does θ_{kv} appear in this product? The number of times v appears in x. This gives our final expression for the probability of document x,

$$p(x \mid \theta_k) = \prod_{v=1}^{V} \theta_{kv}^{n_v(x)}, \tag{13}$$

where $n_v(x)$ is the number of times term v occurred in x.
- This is sometimes called a "Naive Bayes" classifier. (It's a silly name: Most models are naive and there isn't much Bayesian about this one.)

Prediction

- In prediction we are given the parameters, $\theta_{1:K}$ and π, and an unlabeled data point x. We want to predict the label for x.

- We use Bayes rule to compute the posterior distribution of the label

$$P(C \mid x) \propto P(x \mid C)P(C). \tag{14}$$

This is proportional because the denominator $p(x)$ is constant with respect to C.

- For each possible label,

$$p(c \mid x) \propto p(x \mid \theta_c)\pi_c \tag{15}$$

- The precise form of the first term depends on the class-conditional data model.

- In the Gaussian case (with fixed variance) notice that

$$p(c \,|\, x) \propto \left(\frac{1}{2\sigma^2}(x - \mu_c)^2\right) \pi_c. \tag{16}$$

  Everything else is constant with respect to the class label c.

- This equation says that we look at the squared difference between x and each class, weighted by the prior probability of that class.

- In the multinomial case, for each class we consider the probability that x was "generated" by its parameter, weighted again by the prior probability of each class,

$$p(c \,|\, x) \propto \left(\prod_{v=1}^{V} \theta_{cv}^{n_v(x)}\right) \pi_c \tag{17}$$

- In practice, we can take the label of maximum posterior probability. Or we can compute the probabilies and report a distribution.

Fitting

- Now we turn to the problem of finding parameters given data. We will find maximum likelihood estimates of the class conditional distributions $\theta_{1:K}$ and the class proportions π.

- The labeled data set is $\{x_i, c_i\}_{i=1}^{n}$. E.g.,
  - Labeled images
  - Labeled documents
  - Labeled genes
  - Labeled songs

- Taking the log of the product of joint distributions $p(c_i)p(x_i \,|\, c_i)$, the log likelihood is

$$\mathscr{L}(\pi, \theta_{1:K}) = \sum_{i=1}^{n} \sum_{j=1}^{k} c_i^j \log \pi^j + c_i^j \log p(x_i \,|\, \theta_j) \tag{18}$$

- Finding the MLEs of π and $\theta_{1:K}$ decomposes into $K+1$ MLE problems.

- First, the MLE of the class proportions is

$$\hat{\pi} = \arg\max_{\pi} \sum_{i=1}^{n} \sum_{j=1}^{k} c_i^j \log \pi^j \tag{19}$$

  This is simply the empirical proportion of each class

$$\hat{\pi}^j = \frac{\sum_{i=1}^{n} c_i^j}{n}, \tag{20}$$

  where the numerator is the number of times we saw class j.

- The MLE of each class conditional parameter is

$$\hat{\theta}_j = \arg\max_{\theta_j} \sum_{i=1}^{n} c_i^k \log p(x_i \mid \theta_j). \tag{21}$$

  Notice that

  - Only the points labeled with class j play a role in this objective function.
  - This is like taking a simple MLE of those points, drawn IID from θ_j.
  - Operationally, take each class and compute the MLE of its parameter from the data assigned to it. In a Bernoulli case, compute the probability of heads. In a Guassian case, compute the empirical mean and variance.

Example: Simple Gaussian classification

- The data are average RGB values for images.
  - Whole images are summarized in a single color

- Draw the graphical model
  - Write the joint
  - Write the MLEs
  - Write the posterior given a new image

- Show the demo

Example: Multinomial classification

- Given labeled data, we can fit a model and classify new data points.

- This strategy is common in classifying documents. To review:
  - x_i is a collection of word counts.
  - x_i^u is the number of times word u occurred.
  - The class conditional parameter is a point on the term simplex,

$$\begin{aligned} \theta_{ku} &> 0 && (22) \\ \textstyle\sum_{u=1}^{v} \theta_{ku} &= 1. && (23) \end{aligned}$$

  - The class conditional probability is

$$p(x_i^v \mid \theta_k) = \prod_{u=1}^{v} \theta_{ku}^{x_{iu}} \tag{24}$$

  - This assumes that the collection of words came IID from θ_k. (You can confirm this.)

- To complete the algorithm, we only need the MLE of θ_j from the collection of documents.

- This is the proportion of times that each word occurred in each class j document:

$$\hat{\theta}_j^u = \frac{\sum_{i=1}^{n} c_i^j x_i^u}{\sum_{i=1}^{n} c_i^j \sum_{w=1}^{v} x_i^w}. \tag{25}$$

  - Numerator: The number of times word u occurred in documents of class j
  - Denominator: The number of words that occurred in documents of class j

- What happens if a test document contains a word that we never saw in training?

- In text models, we often *smooth* the parameter estimates.
  - The simplest smoother is to add a "pseudocount" to each word (such as one) before computing the MLE.
  - What does this do to the probabilities of frequent words? rare words?
  - Smoothing gets more complicated than that.

- It has interesting connections to

- Bayesian statistics: it can be construed as assuming the distribution came from a prior and then computing the posterior expectation of that distribution
- WWII history: A. Turing and I. J. Good developed smoothing to break the Nazi code.

- TODO: Implementing
  - The log trick