EECS 445: Machine Learning **Fall 2016**

Discussion 8: EM

Written by: Chansoo Lee *Last Updated: December 19, 201612:08 Noon*

8.1 Supervised Naive Bayes

In supervised learning of Naive Bayes, we get *labeled* training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $\mathbf{x}^{(i)} \in \{1, \ldots, M\}^D$ and $y^{(i)} \in \{1, \ldots, C\}$. The estimation (training) problem is to find the maximum log likelihood estimate parameters. The log-likelihood function breaks down into a sum of simple terms, which we can easily differentiate:

$$\sum_{\mathbf{x}, y \in \mathcal{D}} \log P(y, \mathbf{x}; \pi, \theta) = \sum_{\mathbf{x}, y \in \mathcal{D}} \big(\log P(y) + \log P(y|\mathbf{x}) \big) \tag{8.1}$$

$$= \sum_{\mathbf{x}, y \in \mathcal{D}} \left(\log \pi_y + \sum_{i=1}^{D} \log \theta_{y d x_d} \right). \tag{8.2}$$

Because each log term involves only one parameter, when we differentiate the above expression with respect to any model parameter π_c or θ_{cdm}, we have a very simple expression, which we can set it to zero and easily solve for.

The closed-form solution shows that **sufficient statistics** required to solve the above problem are counts: N_c, number of examples in category c and N_{cdm}, number of examples in category c with d-th feature taking value M.

8.2 Unsupervised Naive Bayes

In **unsupervised Naive Bayes**, we get unlabeled training data $\{\mathbf{x}^{(i)}\}_{i=1}^{n}$ where each data is a binary feature vector: $\mathbf{x}^{(i)} \in \{0, 1\}^D$. The estimation problem is to find the maximum log likelihood estimate parameters for our *partially observed data*:

$$\sum_{\mathbf{x} \in \mathcal{D}} \log P(\mathbf{x}; \pi, \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log \left(\sum_{c=1}^{C} P(\mathbf{x}, y = c) \right) \tag{8.3}$$

$$= \sum_{\mathbf{x} \in \mathcal{D}} \log \left(\sum_{c=1}^{C} \left(\pi_c \prod_{d=1}^{D} \theta_{c d x_d} \right) \right). \tag{8.4}$$

Since our probabilstic model does not directly define the marginal distribution over $\mathbf{x}$, we have to marginalize y out from the joint distribution (8.3). As a result, there is now a summation inside the logarithm, which makes derivatives messy. If you want to check, take derivatives with respect to π_1 of (8.2) and (8.4).[1]

The important observation is that we can break this hard problem into two easy problems: (1) If we fix the model parameters, then it is a simple prediction step to compute the conditional distribution of the missing values: $P(y_d = c|\mathbf{x}; \theta, \pi) \propto \pi_c \prod_{d=1}^{D} \theta_{c d x_d}$. (2) If we fill in the missing values, it is the supervised learning problem that we know how to solve.

This leads to the EM algorithm, an alternating optimization technique. EM solves the generic question of: *how do we find the MLE parameters given partially observed data in a principled manner that properly accounts for the dependence among random variables in our probabilstic model?*

[1] An alternative explanation for the hardness of unsupervised setting is that the sufficient statistics to compute the MLE estimates are not provided.

8.3 Soft assignment EM

- Initiliaze the parameters to the values that will break the symmetry, and then repeat:
- In the E-step, we compute the distribution of the missing data. We often say that we compute the *expected sufficient statistics*, because we fill in the missing data only probabilistically.
- In the M-step, update the model parameters to be the MLE based on new expected sufficient statistics.

8.3.1 Case Study: Binary Unsupervised Naive Bayes

Suppose the training set has two data points $\mathbf{x}^{(1)} = (0, 1)$ and $\mathbf{x}^{(2)} = (1, 1)$; for clarity, we note that the dimensions are: $N = 2, C = 2$, $D = 2$, and $M = 2$. In this case study, our binary features and labels take values $\{0, 1\}$ instead of $\{1, 2\}$.

Initilization Suppose our random initialization gives $\pi_c = 0.7, \theta_{111} = 0.9, \theta_{011} = 0.3, \theta_{121} = 0.6, \theta_{021} = 0.2$.

Iteration 1: E-step Compute the probability for each assignment of missing values[2].

$y^{(1)}$	$y^{(2)}$	unnormalized probability	noramlized probability
1	1	$(0.7)(0.1)(0.6)(0.7)(0.9)(0.6) = 0.015876$	0.4773
1	0	$(0.7)(0.1)(0.6)(0.3)(0.3)(0.2) = 0.000756$	0.0227
0	1	$(0.3)(0.7)(0.2)(0.7)(0.9)(0.6) = 0.015876$	0.4773
0	0	$(0.3)(0.7)(0.2)(0.3)(0.3)(0.2) = 0.000756$	0.0227

The table reads as: the missing y value of the i-th example is assigned the value specified in column i, with probability in column 4.

Excercise 8.1. *Compute the expected sufficient statistics using the table.*

Proof: Consider $\overline{N_1}$, the *expected* number of examples with class label $y = 1$:

- row 1 shows with probability 0.4773 there are two instances with $y = 1$.
- row 2 shows with probability 0.0227 there is one instance with $y = 1$.
- row 3 shows with probability 0.4773 there is one instance with $y = 1$.
- row 4 shows with probability 0.0227 there is zero instance with $y = 1$.
- Therefore,
$$\overline{N_1} = (0.4773)(2) + (0.0227)(1) + (0.4773)(1) + (0.0227)(0) = 1.4546.$$
- Because class labels are binary,
$$\overline{N_0} = \overline{N} - \overline{N_1} = 2 - 1.4546 = 0.5454$$

Note that given fully observed data, we would do a simple counting where each example adds either 0 or 1. Each example adds 1 to N_c for a single c. But for the EM, each example adds an a *fraction* between 0 and 1, such that it *distributes* the count 1 across all N_c's.

Consider $\overline{N_{111}}$, the *expected* number of examples with class label $y = 1$ and first feature taking value 1.

- row 1 shows with probability 0.4773, our train set is $((0, 1), 1)$ and $((1, 1), 1)$. So there is one example with $y = 1$ and $x_1 = 1$.

[2]Unnoramlized probability is $P(\mathbf{x}, y)$ and the normalized probability is $P(y|\mathbf{x})$.

- row 2 shows with probability 0.0227, our train set is $((0,1),1)$ and $((1,1),0)$. So there is zero example with $y=1$ and $x_1=1$.
- row 3 shows with probability 0.4773, our train set is $((0,1),0)$ and $((1,1),1)$. So there is one example with $y=1$ and $x_1=1$.
- row 4 shows with probability 0.0227, our train set is $((0,1),0)$ and $((1,1),0)$. So there is zero example with $y=1$ and $x_1=1$.
- Therefore,
$$\overline{N_{111}} = (0.4773)(1) + (0.0227)(0) + (0.4773)(1) + (0.0227)(0) = 0.9546.$$
- Because features are binary, $\overline{N_{110}} = \overline{N_1} - \overline{N_{111}} = 1.4546 - 0.9546 = 0.5$

Similarly, we can compute the rest of the parameters. For example,

$$\overline{N_{121}} = (0.4773)2 + (0.0227) + (0.4773) = 1.4546$$ ■

Iteration 1: M-step It's the same MLE parameter computation as in supervised Naive Bayes, except we use *expected* counts instead of counts.

$$\pi_1 = \frac{\overline{N_1}}{N} = 1.4546/2 = 0.7273.$$

$$\theta_{111} = \frac{\overline{N_{111}}}{\overline{N_1}} = 0.6563.$$

$$\theta_{121} = \frac{\overline{N_{121}}}{\overline{N_1}} = 1.$$

Efficient E-step The presented implementation of E-step takes exponential time $\Omega(N^C)$, because it computes the full joint distribution over N labels. Indeed, you might have noticed a bit of redundancy when we computed the expected counts.

The trick is that we can compute the distribution of the unobserved variables indepnedently, making it $O(NC)$ computation. Mathematically,

$$\begin{aligned}\overline{N_1} &= P(y^{(1)}=1,y^{(2)}=1) + P(y^{(1)}=1,y^{(2)}=1) + P(y^{(1)}=1,y^{(2)}=0) + P(y^{(1)}=0,y^{(2)}=1)\\ &= (P(y^{(1)}=1,y^{(2)}=1) + P(y^{(1)}=1,y^{(2)}=0)) + (P(y^{(1)}=1,y^{(2)}=1) + P(y^{(1)}=0,y^{(2)}=1))\\ &= P(y^{(1)}=1) + P(y^{(2)}=1)\end{aligned}$$

Then we compute the probability distributions for missing values, independently for each example:

$$P(y^{(1)}=1) = \frac{(0.7)(0.1)(0.6)}{(0.7)(0.1)(0.6)+(0.3)(0.7)(0.2)} = 0.5$$

$$P(y^{(2)}=1) = \frac{(0.7)(0.9)(0.6)}{(0.7)(0.9)(0.6)+(0.3)(0.3)(0.2)} = 0.9546$$

It is also easy to note that

$$\overline{N_0} = 2 - \overline{N_1} = P(y^{(1)}=0) + P(y^{(2)}=0).$$

This gives a very intuitive explanation of the EM algorithm on unsupervised Naive Bayes. Given labeled (fully observed) data, we replace the probabilities with indicator functions; each example contributes one count to either N_1 or N_0. Given unlabeled (partially observed) data, each instance *distributes* a total of one count over N_1 and N_0.

Excercise 8.2. *Numerically work out the second iteration of EM, using the efficient E-step.*

8.3.2 Hard assignment EM

Sometimes it is difficult (due to mathematical complexity or limited computation time) to handle fractional contribution of each data point. In such applications, we can make E-step assign the missing values with the highest probability, breaking ties arbitrarily. M-step becomes the MLE given fully observed data.

Excercise 8.3. *Show that k-means is equivalent to the hard-assignment EM for GMM clustering with fixed identiyy covariance matrices.*

Excercise 8.4. *What are the differences between soft and hard assignment EM?*

The hard-assignment EM explores the combinatorial space of missing variable assignments. The soft-assignment EM, on the other hand, explores the continuous space.

In clustering, the hard assignment EM tends to amplify the contrast among classes, while soft assignment EM attempts to model mixed-class memberships.

8.3.3 Random assignment EM

In E-step, we compute the distribution over missing values and sample accordingly, instead of computing the expected statistics. Use the random samples to fill in the missing data. M-step becomes the MLE given fully observed data.