Week 4: Naïve Bayes

Instructor: Sergey Levine

1 Naïve Bayes Models: Recap and Definitions

In this section, we'll recap and formalize the discussion from last week. The ideas are the same, but we'll go into a little bit more depth. In the naïve Bayes model, we have a dataset that consists of discrete labels $y \in \{0, \dots, L_y - 1\}$ and attributes $\{x_1, \dots, x_K\}$, where each $x_k \in \{0, \dots, L_k - 1\}$. We use $\mathbf{x}$ to denote the vector $(x_1, \dots, x_K)^T$. Note that y and $\mathbf{x}$ here are *random variables.* The values of those random variables are referred to as $\mathbf{x}^i$ and y^i, where the dataset is $\mathcal{D} = \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$. We primarily discussed binary variables last week, but the generalization to multinomials is straightforward. The objective in naïve Bayes is the likelihood

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p(\mathbf{x}^i, y^i|\theta) = \sum_{i=1}^{N} \left[\log p(y^i|\theta) + \sum_{k=1}^{K} \log p(x_k^i|y^i, \theta) \right].$$

This is sometimes referred to as a *generative* objective, because it models how the labels y give rise to (or generate) the attributes x_k. For example, in the case of the rain storm discussed last week, the probabilistic model assumes that the storm gives rise to rain, lightening, and clouds. This is in contrast to a *discriminative* objective of the form $p(y|\mathbf{x})$, which does not aim to model how the features or attributes are influenced by the label, but vice-versa.

In naïve Bayes, we typically use a disjoint subset of the parameters θ for each probability table (sometimes referred to as a "factor"): we have one set of parameters for the prior $p(y)$, another set for $p(x_1|y = 0)$, another for $p(x_1|y = 1)$, another for $p(x_2|y = 0)$, etc. We can write the set of parameters as: $\theta = \{\theta_y, \theta_{x_1,0}, \dots, \theta_{x_1,L_y}, \dots, \theta_{x_K,0}, \dots, \theta_{x_K,L_y}\}$. Note that each entry in this set might itself be one or more values. In the case where all variables are binary, there is one parameter per probability table. But if we have multinomials, then each e.g. $\theta_{x_k,0}$ might itself be a vector.

Indeed, the standard way to fit naïve Bayes models is to directly determine the parameters of the multinomial distribution, which correspond to each entry in each table $p(x_k = \ell|y = j)$. Let's first check our understanding:

Question. How many probability tables are there, if we have K attributes and L_y labels?

Answer. We have one probability table for each attribute and each value of the label, plus one more for the prior $p(y)$, so the total is $KL_y + 1$. The set of tables for a single attribute (for all labels y) is sometimes referred to as a conditional probability table (CPT), and we have K of those.

To fit each CPT, we can simply count the number of occurrences of each pair $(x_k = \ell, y = j)$ in the dataset and normalize to make sure the distribution adds up to 1 across all values of x_k. So we have:

$$p(x_k = \ell | y = j, \theta_{x_k,j}) = \frac{\text{Count}(x_k = \ell \text{ and } y = j)}{\sum_{\ell'} \text{Count}(x_k = \ell' \text{ and } y = j)}.$$

Fitting the prior is even easier:

$$p(y = j|\theta_y) = \frac{\text{Count}(y = j)}{\sum_{j'} \text{Count}(y = j')}.$$

Note that we can easily derive this update rule from the log-likelihood by noting that each set of parameters affects only one probability table. First, let's be explicit about which part of the likelihood depends on which parameter:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p(\mathbf{x}^i, y^i|\theta) = \sum_{i=1}^{N} \left[\log p(y^i|\theta_y) + \sum_{k=1}^{K} \log p(x_k^i|y^i, \theta_{x_k,y^i}) \right].$$

Now, let's look at the gradient with respect to $\theta_{x_{k'},j'}$:

$$\frac{d}{d\theta_{x_{k'},j'}} \sum_{i=1}^{N} \left[\log p(y^i|\theta_y) + \sum_{k=1}^{K} \log p(x_k^i|y^i, \theta_{x_k,y^i}) \right] =$$

$$\sum_{i=1}^{N} \left[\underbrace{\frac{d}{d\theta_{x_{k'},j'}} \log p(y^i|\theta_y)}_{\text{always zero}} + \sum_{k=1}^{K} \underbrace{\frac{d}{d\theta_{x_{k'},j'}} \log p(x_k^i|y^i, \theta_{x_k,y^i})}_{\text{zero if } k \neq k' \text{ or } y^i \neq j'} \right] =$$

$$\sum_{i=1}^{N} 1_{y^i=j'} \frac{d}{d\theta_{x_{k'},j'}} \log p(x_{k'}^i|j', \theta_{x_{k'},j'}).$$

From here, we can see that only records for which $y = j'$ matter, and among those records, the only part of the log likelihood that matters is $\log p(x_{k'}^i|j', \theta_{x_{k'},j'})$. That means that we can fit each distribution completely separately, and still optimize the log likelihood.

2 Continuous Features

In practice, we might encounter problems where some attributes x_k are discrete and some are continuous. Note that in the naïve Bayes formulation, the CPT for each x_k can be parameterized differently. In general, we call it a conditional probability distribution (CPD) in cases where it cannot be represented in a simple tabular form. This could be any distribution type we want.

Question. If we have an attribute $x_k \in \mathbb{R}$, what kind of distribution could we use for $p(x_k|y)$?

Answer. For real-valued attributes, a Gaussian distribution is often a good choice. So we can set

$$p(x_k|y=j) = \frac{1}{\sigma_{k,j}\sqrt{2\pi}} e^{-\frac{(x_k-\mu_{k,j})^2}{2\sigma_{k,j}^2}}.$$

Note that the mean $\mu_{k,j}$ and variance $\sigma_{k,j}^2$ depends both on the index of the attribute k *and* on the label j: for each label of y, we can have a different mean and variance, just like we have different entries in the CPT for discrete multinomial distributions.

To estimate the parameters $\mu_{k,j}$ and $\sigma_{k,j}^2$, we simply consider the values of x_k^i for all records for which $y^i = j$:

$$\mu_{k,j} = \frac{1}{\text{Count}(y=j)} \sum_{i=1}^{N} \delta(y^i = j) x_k^i$$
$$\sigma_{k,j}^2 = \frac{1}{\text{Count}(y=j)} \sum_{i=1}^{N} \delta(y^i = j)(x_k^i - \mu_{k,j})^2,$$

where $\delta(\dots)$ is the delta function, which is 1 if the argument is true, and 0 otherwise. Again, we can show that this is the optimal solution for $\mu_{k,j}$ and $\sigma_{k,j}^2$ by differentiating the log likelihood and solving for the solution with a derivative of zero. We can also impose a prior on the parameters as before.

3 Bayesian Networks Intro

Naïve Bayes is a special case of a type of model called a Bayesian network. We will not cover Bayesian networks in detail in this course, but I will introduce the idea briefly. Bayesian networks generalize the idea behind naïve Bayes to model distributions over groups of variables with more complex conditional independence relationship. A Bayesian network consists of a collection of CPDs, such that the product of the CPDs is a full joint distribution over all the variables that the Bayesian network models. Some of these may be labels and some may be attributes: we don't have to make the distinction during training, because they are all treated the same way. We can visualize Bayesian networks graphically by drawing a graph where each variable is a node, and a directed edge indicates that the target node depends on the source node. So, for example, if we have a graph over variables x_1, x_2, x_3 with an edge from x_1 to x_3 and from x_2 to x_3, that corresponds to the following factorization of the joint:

$$p(x_1, x_2, x_3) = p(x_3|x_1, x_2)p(x_1)p(x_2).$$

Note that any node without an incoming edge simply gets a prior (e.g. $p(x_1)$), while nodes that have incoming edges are conditioned on all of the nodes from which these edges originate. As with naïve Bayes, we can learn the parameters of a Bayesian network independently for each CPD. For example, to fit $p(x_3 = \ell | x_1 = j, x_2 = k)$, we can set:

$$p(x_3 = \ell | x_1 = j, x_2 = k) = \frac{\text{Count}(x_3 = \ell \text{ and } x_1 = j \text{ and } x_2 = k)}{\text{Count}(x_1 = j \text{ and } x_2 = k)}.$$

Things get a bit more complex when conditioning on continuous variables, and there we typically have to employ parameterized distributions such as linear Gaussians (like we did in linear regression).

Typically, once we learn the parameters of a Bayesian network, we might like to perform inference at test time to infer the value of an unknown variable. For example, we can draw a complex graph of a patient's symptoms that describes their actual independence structure and relationship. Then at test time, we can measure some of the symptoms (but not others), and we can infer the rest by choosing the values that maximize the probability of the final joint distribution.

Note, however, that unlike with naïve Bayes, inference in Bayesian networks can become very difficult. For example, say we fit a Bayesian network to K variables $x_1, \dots, x_K$, and at test time $M < K$ of these variables are unknown. Even if each variable is binary, there are 2^M possible assignments. In general, once we have more than one unknown (like we did in naïve Bayes), the simple brute force inference method becomes intractable. So a crucial ingredient for working with large Bayesian networks is tractable and efficient inference algorithms.