SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

01.112 Machine Learning, Fall 2019
Lecture Notes from Week 6

12. Generative Models, Naive Bayes

Last update: Tuesday 22nd October, 2019 14:08

1 Generative Models

So far we have primarily focused on classification, where the goal is to find a separator that sets apart two classes. The internal structure of the classes is not directly captured by (or cared for) the classifier. In fact, datasets with very different structure may have exactly the same separator. Intuitively, understanding the structure of the data should be helpful for classification as well. We have touched this issue briefly in the context of k-means clustering. However, clustering provided a limited way of capturing the properties of input examples. For example, simply providing a cluster representative shows nothing about how the points are spread. Moreover, not all data have clusters of the type that k-means is geared to find.

An example of such a dataset is shown in Figure 1.

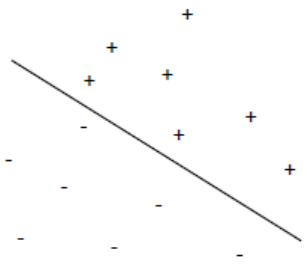

Figure 1: A labeled dataset where the class distinction cuts across a single cluster

Today, we will discuss using probabilistic models for data. We will demonstrate how these models can be directly used in classification: once we have a model for each class of examples, we can estimate how likely it was generated by each class-specific model, and determine the predicted label accordingly. A central question in probabilistic modeling is how to select a model that best fits the properties of the data we are trying to capture. We will describe several models and their properties.

- **Gaussians** Consider a cluster of points shown in Figure 2. To summarize this data, the model should capture (1) the center of the group (mean); (2) how spread the points are from the

center. The simplest model, the one that makes fewest assumptions above and beyond the mean and the spread, is a Gaussian:

$$P(x; \mu, \sigma) = C \cdot e^{-\frac{1}{2\sigma^2}\|x-\mu\|^2}$$

The value of σ determines how the probability of seeing points away from the mean decays. A small σ will result in a tight cluster of points (points close to the mean), while a large value of σ corresponds to a widely spread cluster of points.

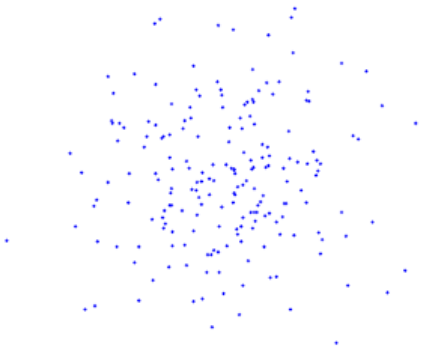

Figure 2: A cluster of points.

- **Multinomials** We can use probabilistic models with data other than real valued points in $\mathcal{R}^d$. Consider, for example, documents represented by bags-of-words. While in supervised classification we mapped each document into a point in space, here we will look at a different representation: we will treat text as a bag of words to be *generated*. Our model will generate different bags of words with different probability. We are looking for a model that generates the observed bags (i.e., documents in our dataset) with high probability.

Naive Bayes

To make our task simpler, we will further simplify the setting by assuming that words in a document are generated one word at a time, each word independently from others, but with a specific distribution. To specify such a model, each word $w \in W$ is associated with parameter θ_w

$$\begin{aligned} \theta_w &\geq 0 \\ \sum_{w \in W} \theta_w &= 1 \\ P(w|\theta) &= \theta_w \end{aligned}$$

For example, consider two probability distributions over four words: dog, cat, tulip, rose. The first distribution has parameters $\theta_{dog} = 0.5$, $\theta_{cat} = 0.4$, $\theta_{tulip} = 0.05$, $\theta_{rose} = 0.05$, while the

second one is parametrized as $\theta_{dog} = 0.1$, $\theta_{cat} = 0.1$, $\theta_{tulip} = 0.5$, $\theta_{rose} = 0.3$. Now consider the following document (i.e., bag of words): *dog, dog, cat, dog, cat, tulip*. Which distribution is more likely to generate this document? To answer this question, we need to compute the likelihood of the document $D = \{w_1, w_2, \ldots, w_n\}$ under a given parameter setting θ:

$$P(D|\theta) = \prod_{i=1}^{N} \theta_{w_i} = \prod_{w} \theta_w^{n(w)}$$

where $n(w)$ is a count of w in document D.

In oder to use these probabilistic models, we need a way to estimate the parameters. Let's start with the case of a single document. We want to find θ that maximize the likelihood of observed data

$$\max_{\theta} P(D|\theta) \Leftrightarrow \max_{\theta} \log P(D|\theta)$$
$$\log P(D|\theta) = \log \prod_{w} \theta_w^{n(w)} = \sum_{w} n(w) \log \theta_w$$

We need to find θ that maximizes this expression. To see how this works, let's make it even simpler and assume that our vocabulary is of size 2, i.e., $W = \{0, 1\}$. Then

$$\log P(D|\theta) = n(0) \log \theta_0 + n(1) \log \theta_1 = n(0) \log(1 - \theta_1) + n(1) \log \theta_1$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_1}(n(0) \log(1 - \theta_1) + n(1) \log \theta_1) &= 0 \\
-\frac{n(0)}{1 - \theta_1} + \frac{n(1)}{\theta_1} &= 0 \\
-n(0)\theta_1 + n(1)(1 - \theta_1) &= 0 \\
-n(0)\theta_1 + n(1) - n(1)\theta_1 &= 0 \\
-\theta_1(n(0) + n(1)) &= -n(1) \\
\hat{\theta}_1 &= \frac{n(1)}{n(0) + n(1)}
\end{aligned}$$

Using Lagrange multipliers, you can show that parameters will have a similar form for an arbitrary size vocabulary:

$$\hat{\theta}_w = \frac{n(w)}{\sum_{w' \in W} n(w')}$$

In the discussion above, we considered training from a single document. Similar argument can be applied to the case where we are estimating parameters from multiple documents. Due to our

independence assumption, the documents are "collapsed" into a giant bag, so we compute word counts across the documents.

$$P(D_1, \ldots, D_T|\theta) = \prod_{t=1}^{T} P(D_t|\theta) = \prod_{w \in W} \theta_w^{\sum_{t=1}^{T} n_t(w)}$$

Let us put the generative model in use for classification problems. We will start with the task of document classification. We are given two sets of documents, with known labels "+" and "-". We are interested in a model that generates documents for each class. Those are *class-conditional* distributions:

$$P(D|\theta^+) = \prod_w (\theta_w^+)^{n(w)}$$
$$P(D|\theta^-) = \prod_w (\theta_w^-)^{n(w)}$$

We estimate θ_w^+ and θ_w^- from the corresponding labeled documents using the method described above. Now, we have two multinomials, each tailored to generate one class of documents. Given a document with an unknown label, how can we classify it? Assume that the likelihood of "+" and "-" is the same (later, we will remove this assumption). We will select the label based on which model assigns a higher likelihood to the document:

$$\log \frac{P(D|\theta^+)}{P(D|\theta^-)} = \begin{cases} \geq 0, & + \\ < 0, & - \end{cases}$$

where the log-likelihood ratio as a discriminant function can be written as

$$\begin{aligned} \log \frac{P(D|\theta^+)}{P(D|\theta^-)} &= \log P(D|\theta^+) - \log P(D|\theta^-) \\ &= \sum_w n(w)(\log \theta_w^+ - \log \theta_w^-) \\ &= \sum_w n(w) \underbrace{\log \frac{\theta_w^+}{\theta_w^-}}_{=\theta_w} \end{aligned}$$

This expression represents a linear classifier in a feature representation that concatenates word counts:

$$\log \frac{P(D|\theta^+)}{P(D|\theta^-)} = \Phi(D) \cdot \theta = \begin{bmatrix} n(w_1) \\ \vdots \\ n(w_{|W|}) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{|W|} \end{bmatrix} \tag{1}$$

Now let's consider the case where the likelihood of the label classes is not the same. We can add the prior class frequencies, $P(y = +)$ and $P(y = -)$ directly to the log-likelihood ratio, i.e.,

$$\log \frac{P(D|\theta^+)P(y=+)}{P(D|\theta^-)P(y=-)} = \sum_w n(w) \underbrace{\log \frac{\theta_w^+}{\theta_w^-}}_{\theta_w} + \underbrace{\log \frac{P(y=+)}{P(y=-)}}_{=\theta_0} = \sum_w n(w)\theta_w + \theta_0$$

or, equivalently, classify according to the posterior probability

$$P(y=+|D) = \frac{P(D|y=+)P(y=+)}{P(D)} = \frac{P(D|y=+)P(y=+)}{P(D|y=+)P(y=+) + P(D|y=-)P(y=-)}$$

where $P(D|y=+) = P(D|\theta^+) = \prod_w (\theta_w^+)^{n(w)}$. Once we estimated θ_w^+ and θ_w^- from the training data, we can classify an unlabeled document D by comparing the values of $P(y=+|D)$ and $P(y=-|D)$.

As we can see, such a model makes very simple strong independence assumptions. Such a model is known as the *Naive Bayes* model.

Smoothing

One important issue related to parameter estimation is accounting for unseen events. Consider what happens if our training sample does not include word w? In this case, $\hat{\theta}_w = 0$. This problem will be particularly acute if we have little training data and many words are unseen during training.

One way to deal with the the sparsity problem is to introduce a prior distribution over parameters θ:

$$P(\theta|\lambda_1, \ldots, \lambda_n) = P(\theta|\lambda) = C \cdot \prod_w \theta_w^{\lambda_w}$$

where $\lambda_1, \ldots, \lambda_n$ are hyper-parameters. θ that maximizes this prior is simply $\hat{\theta}_w = \lambda_w / \sum_{w'} \lambda_{w'}$ and represents the "default" value that the prior highlights. During estimation, the hyper-parameters $\lambda_1, \ldots, \lambda_n$ act as *pseudo-counts* that supplement the observed counts in the data:

$$P(\theta|D) \propto P(D|\theta)P(\theta|\lambda) = \prod_w [\theta_w^{n(w)} \cdot \theta_w^{\lambda_w}] = \prod_w \theta_w^{n(w)+\lambda_w}$$

In other words, when we find the parameter values that maximize the posterior $P(\theta|D)$, we simply combine the observed counts with the pseudo counts from the prior. When observed counts are much larger than the hyper-parameters, the effect of the prior is miniscule. However, if the dataset is small, the pseudo counts will push the parameter estimates towards the default values specified by the prior.

Learning Objective

You need to know:

1. What is a generative model and how it is different from a discriminative model.
2. How to learn model parameters for a generative model that involve multinomial distributions.
3. How to make predictions using a learned generative model.