Naive Bayes

→ Feature vectors x are discrete-valued.

Motivating Example: Spam filter → Spam / Not Spam

⇒ Will begin our construction of our spam filter by specifying the features x_i used to represent an email.

⇒ We will represent an email via a feature vector whose length is equal to the number of words in the dictionary.

↳ If an email contains the i^{th} word of the dictionary, then we will set $x_i = 1$; otherwise, we let $x_i = 0$.

⇒ The set of words encoded into the feature vector is called the Vocabulary, so the dimension of x is equal to the size of vocabulary.

⇒ If Vocabulary = 50000 words, $x \in \{0,1\}^{50,000}$.

↳ If we were to model x explicitly with a multinomial distribution over the 2^{50000} possible outcomes, then we'd end up with a $(2^{50,000}-1)$ dimensional parameter vector.

↳ This is clearly too many parameter.

⇒ To model $P(x|y)$, we will therefore make a very strong assumption

↳ We will assume that the x_i's are Conditionally independent given y.

⇒ This assumption is called the Naive Bayes (NB) Assumption.

⇒ The resulting algorithm is called the Naive Bayes classifier.

$$P(x_1, \cdots x_{50000} | y)$$

$$= P(x_1|y)\, P(x_2|y, x_1)\, P(x_3|y, x_1, x_2) \cdots P(x_{50,000}|y, x_1, x_2 \cdots x_{49999})$$

$$= P(x_1|y)\, P(x_2|y) \cdots P(x_{50000}|y)$$

$$= \prod_{i=1}^{m} P(x_i|y)$$

⇒ Even though the Naive Bayes assumption is an extreamly strong assumptions, the resulting algorithm works well on many problem.

⇒ Our model is parameterized by $\phi_{i|y=1}$, $\phi_{i|y=0}$ & ϕ_y

$$\phi_{i|y=1} \to P(x_i=1|y=1) \qquad \phi_{i|y=0} \to P(x_i=1|y=0) \qquad \phi_y \to P(y=1)$$

⇒ Given the training set $\{(x^{(i)}, y^{(i)}) ; i=1, \cdots m\}$

We can write joint likelyhood of the data:

$$\mathcal{L}(\phi_y, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} P(x^{(i)}, y^{(i)})$$

⇒ Maximizing this with respect to ϕ_y, $\phi_{i|y=0}$ & $\phi_{i|y=1}$ gives the maximum likelihood estimates:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

⇒ To make a prediction on a new example with features x, we then simply calculate:

$$P(y=1|x) = \frac{P(x|y=1)\, P(y=1)}{P(x)}$$

$$= \frac{\left(\prod_{i=1}^{n} P(x_i|y=1)\right) P(y=1)}{\left(\prod_{i=1}^{n} P(x_i|y=1)\right) P(y=1) + \left(\prod_{i=1}^{n} P(x_i|y=0)\right) P(y=0)}$$

⇒ and pick whichever class have the higher Posterior probability.

⇒ Generalization of Naive Bayes where x_i can take values in $\{1, 2, \cdots K_i\}$ is Straight forward.

↳ Here we simply model $P(x_i|y)$ as multinomial rather than bernoulli.

⇒ When the original, continuous-valued attributes are not well modeled by a multivariate normal distribution, descretizing the feature and using Naive Bayes (instead of GDA) will often result in a better classifier.

★ <u>Laplace smoothing</u>

⇒ Statistically its a bad idea to estimate the probability of some event to be zero just because you haven't seen it before in your finite training set.

⇒ Take the problem of estimating the mean of multinomial random variable z taking values in $\{1, \cdots K\}$.

⇒ We can parameterize our multinomial with
$$\phi_i = P(z = i)$$

⇒ Given a set of m independent observations $\{z^{(1)}, \cdots z^{(m)}\}$, the maximum likely estimate are given by
$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}{m}$$

⇒ If we use maximum likelihood estimates, then some of the ϕ_j's might end up zero.

⇒ To avoid this we can use Laplace smoothing which replaces the above estimate with

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} + 1}{m + K}$$

⇒ Returning back to our Naive Bayes classifier, with Laplace smoothing we therefore obtain the following estimate of the parameters:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}$$

★ Event models for text Classification

⇒ Naive Bayes works well for many classification problems, for text classification there is a related model that does even better.

⇒ Naive Bayes as presented uses

{ Multi-Variate Bernoulli event model }

⇒ Here's a different model, called the multinomial event model

⇒ We will use a different notation and set of features for representing emails.

⇒ Let x_i denotes the identity of the ith word in the email.

$$x_i \in \{1, \cdots |V|\}$$

where $|V| \rightarrow$ Size of Vocabulary

⇒ An email of n words is now represented by a vector $(x_1, x_2, \cdots x_n)$ of length n.

↳ n can vary for different documents.

⇒ In the multinomial event model, we assume that the way an email is generated is via a random process:

① → Spam / non-Spam is first determined

② → Then, the Sender of the email writes the email by first generating x_1 from some multinomial distribution over words $P(x_1|y)$

③ → Next the second word is x_2 is chosen independently of x_1 but from the same multinomial distribution, and Similarly for x_3, x_4, and so on.

④ ↳ Thus the overall probability of a message is given by $P(y)\prod_{i=1}^{n} P(x_i|y)$

⇒ The parameter of our new model are:

$$\phi_y = P(y)$$

$$\left.\begin{aligned}\phi_{i|y=1} &= P(x_j = i \mid y=1)\\ \phi_{i|y=0} &= P(x_j = i \mid y=0)\end{aligned}\right\} \text{ for any } j$$

⇒ If we are given a training set

$\{(x^{(i)}, y^{(i)}); i = 1, \dots m\}$

where $x^{(i)} = (x_1^{(i)}, x_2^{(i)} \dots x_{n_i}^{(i)})$

⇒ The likelyhood of the data is given by

$$\mathcal{L}(\phi, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} P(x^{(i)}, y^{(i)})$$

$$= \prod_{i=1}^{m}\left(\prod_{j=1}^{n_i} P(x_j^{(i)} | y; \phi_{i|y=0}, \phi_{i|y=1})\right) P(y^{(i)}; \phi_y)$$

⇒ Maximizing this yields the maximum likelihood estimates of the parameters:

⇒ Maximizing this yields the maximum likelihood estimates of the parameters:

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)}=k \wedge y^{(i)}=1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)}=1\} n_i + |V|}$$

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)}=k \wedge y^{(i)}=0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)}=0\} n_i + |V|}$$

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{m}$$

{Laplace Smoothing Added}

⇒ While not necessarily the very best classification algorithm, the Naive Bayes classifier often works surprisingly well.

↳ It is often also a very good "first thing to try" given its simplicity & ease of implementation.