Machine Learning

"Field of study that gives computers the ability to learn with out being explicitly programmed" - Arthur Samuel

1. Naive Bayes algorithm

To build the intuition around the Naive Bayes algorithm, we can start by looking at an example. Suppose we wanted to build a spam classifier.

- We will have a vocabulary D words $D = [a, \ldots, ciao, \ldots.]$
- $y \in \{0, 1\}$ (not spam, spam)
- $x \in \mathbb{R}^d, x = \{0, 1\}^D$

An e-mail message could be represented as $x_{ex} = (0, 0, 0, 0, 0, 1, \ldots, 1, 0)$

The joint probability of x and y is

$$P(x, y)) = P(y)P(x \mid y)$$

We can also apply the chain rule of probability

$$P(x \mid y) = P(x_1 \mid y) \cdot P(x_2 \mid x_1, y) \cdot \ldots \cdot P(x_d \mid x_1, \ldots, x_{D-1}, y)$$

That effectively describes the probability of the presence of the i-th word.

Let's now assume the independence of the probability of the words from each other (Naive), and we obtain

$$= P(x_1 \mid y) \cdot P(x_2 \mid y) \cdot \ldots \cdot P(x_D \mid y) = \prod_{i=1}^{D} P(x_j \mid y)$$

If $y \in \{0, 1\}$, then we will use Bernoulli distribution to define the likelihood function L.

Let's summarize the parameters and with respect to their probability function

ϕ_y	$P(y)$	1 parameter to model the prior probability
$\phi_{j \mid y=1}$	$P(x_j \mid y = 1)$	D terms
$\phi_{j \mid y=0}$	$P(x_j \mid y = 0)$	D terms

The process can be practically seen as a $D+1$ coin tosses, the first to bias a message to be spammy or not, the other D referring to the modeled mail (x vector). So, let's find the log likelihood expression, its gradient, and by maximizing it finding the closed form expressions (that exist in this case, if not it should be resolved numerically)

$$L\left(\phi_y, \phi_{j\,|\,y=1}, \phi_{y\,|\,y=0}\right) = \prod_{i=1}^{n} P\left(x^{(i)}, y^{(i)}\right) = \prod_{i=1}^{n} P\left(y^{(i)}\right) P\left(x^{(i)} \,\middle|\, y^{(i)}\right)$$

$$\log L\left(\phi_y, \phi_{j\,|\,y=1}, \phi_{y\,|\,y=0}\right) = l\left(\phi_y, \phi_{j\,|\,y=1}, \phi_{y\,|\,y=0}\right) = \sum_{i=1}^{n} \left[\log P\left(y^{(i)}\right) + \sum_{i=1}^{D} \log P\left(x^{(i)} \,\middle|\, y^{(i)}\right)\right]$$

And by imposing $\nabla l = 0$ we obtain the following closed form non-smoothing expressions

- $\hat{\phi}_y = \dfrac{\sum_{i=1}^{n} I\left[y^{(i)} = 1\right]}{n}$
- $\hat{\phi}_{j\,|\,y=1} = \dfrac{\sum_{i=1}^{n} I\left[x_j^{(i)} = 1 \,\wedge\, y^{(i)} = 1\right]}{\sum_{i=1}^{n} I\left[y^{(i)} = 1\right]}$
- $\hat{\phi}_{j\,|\,y=0} = \dfrac{\sum_{i=1}^{n} I\left[x_j^{(i)} = 1 \,\wedge\, y^{(i)} = 0\right]}{\sum_{i=1}^{n} I\left[y^{(i)} = 0\right]}$

Which means, (2, numerator) if the $x_j^{(i)}$ word appears and the email is spammy $y^{(i)} = 1$, then parameter equals 1 (if not 0), divided by the bias subset. The expression refers to actually counting the examples where this is valid in order to make predictions.

From Bayes theorem

$$P(y = 1 \,|\, x) = \frac{P(x \,|\, y = 1) P(y = 1)}{P(x)} = \frac{P(x \,|\, y = 1) P(y = 1)}{P(x \,|\, y = 1) P(y = 1) + P(x \,|\, y = 0) P(y = 0)}$$

So the probability of a message being spammy given x will look like this

$$P(y = 1 \,|\, x) = \frac{\prod_{j=1}^{D} \phi_{j\,|\,y=1}^{x_j} \left(1 - \phi_{j\,|\,y=1}^{1-x_j}\right) \left(\hat{\phi}_y\right)}{\prod_{j=1}^{D} \phi_{j\,|\,y=1}^{x_j} \left(1 - \phi_{j\,|\,y=1}^{1-x_j}\right) \left(\hat{\phi}_y\right) + \prod_{j=1}^{D} \phi_{j\,|\,y=0}^{x_j} \left(1 - \phi_{j\,|\,y=0}^{1-x_j}\right) \left(\hat{\phi}_y\right)}$$

2. Laplace smoothing

Worth mentioning is the problem of rare words. If a rare words happens to occur in a spam text just once among all the examples, $\phi_{j \mid y=1} = 1$ and $\phi_{j \mid y=0} = 0$, then one 0 can erase overwhelming evidence. Laplace smoothing is a powerful technique that allows us to avoid this problem, by “gaining resilience” to those extreme cases.

Given $\left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{n}$ examples, we add to each one 2 “virtual” examples for them to be spammy or not

$$\left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{n} + \begin{pmatrix}[1, \ldots, 1], 0 \\ [1, \ldots, 1], 1\end{pmatrix} + \begin{pmatrix}[0, \ldots, 0], 0 \\ [0, \ldots, 0], 1\end{pmatrix}$$

Smoothed parameters look like this

- $\hat{\phi}_{y} = \dfrac{1 + \sum_{i=1}^{n} I\left[y^{(i)} = 1\right]}{2 + n}$
- $\hat{\phi}_{j \mid y=1} = \dfrac{1 + \sum_{i=1}^{n} I\left[x_j^{(i)} = 1 \wedge y^{(i)} = 1\right]}{1 + \sum_{i=1}^{n} I\left[y^{(i)} = 1\right]}$
- $\hat{\phi}_{j \mid y=0} = \dfrac{1 + \sum_{i=1}^{n} I\left[x_j^{(i)} = 1 \wedge y^{(i)} = 0\right]}{1 + \sum_{i=1}^{n} I\left[y^{(i)} = 0\right]}$

3. Bernoulli event model

Generalization of our example. We used a *Bernoulli event model* where

$$x \in \mathbb{R}^d,\ x \sim \{0,1\}^d$$

Multinomial event model: if a word occurs more times, the Lshould be higher for spammy.

Given a text example $x^{(i)} = \left(x_1^{(i)}, \ldots, x_j^{(i)}, \ldots, x_d^{(i)}\right)$

$$P(x,y) = P(y)P(x \mid y) = P(y)\prod_{i=1}^{d} P\left(x_j^{(i)} \,\middle|\, y\right)$$

Parameters:

$$L\left(\phi_y, \phi_{k \mid y=0}, \phi_{k \mid y=1}\right) = \prod_{i=1}^{n} P\left(x^{(i)}, y^{(i)}\right) = \prod_{i=1}^{n}\left(\prod_{j=1}^{d} P\left(x_j^{(i)} \,\middle|\, y^{(i)}\right)\right) P\left(y^{(i)}\right)$$

and $\log \nabla L$ (=0 closed forms, $\neq$0 numerical solution using gradient ascent).

- $\hat{\phi}_y = \dfrac{\sum_{i=1}^{n} I\left[y^{(i)} = 1\right]}{n}$
- $\hat{\phi}_{k \mid y=1,0} = \dfrac{\sum_{i=1}^{n}\sum_{i=1}^{d} I\left[x_j^{(i)} = 1 \ \wedge\ y^{(i)} = 1,0\right]}{\sum_{i=1}^{n} I\left[y^{(i)} = 1,0\right]}$

Where the first sum refers to the totality of messages, the second sum the totality of words per message. We are counting the frequency of a word being inside of a spammy class.

Laplace smoothing is applicable

- $\hat{\phi}_y = \dfrac{1 + \sum_{i=1}^{n} I\left[y^{(i)} = 1\right]}{2 + n}$
- $\hat{\phi}_{k \mid y=1,0} = \dfrac{1 + \sum_{i=1}^{n}\sum_{i=1}^{d} I\left[x_j^{(i)} = 1 \ \wedge\ y^{(i)} = 1,0\right]}{|D| + \sum_{i=1}^{n} I\left[y^{(i)} = 1,0\right] d_i}$

Where $|D|$ is the total number of words.

It’s all about counting and taking ratios.

4. Kernel methods

Kernel methods identify a technique applicable to many models. Let's first introduce the concept of *feature maps*.

$$x \in \mathbb{R}^d \rightarrow \phi(x) \in \mathbb{R}^p$$

A feature map is a function that takes an example (independent variable) as input and outputs a p-long vector, where usually $p >> d$ (p can be *uncountably infinite*).

Let's now apply the feature map to known models and procedures

Linear regression's hypothesis function	$h_\theta(x) = \theta^T x$	$h_\theta(x) = \theta^T \phi(x)$	$x \in \mathbb{R}^d$ $\phi(x) \in \mathbb{R}^p$ $\theta \in \mathbb{R}^{d,p}$
Linear regression's GAUR	$\theta = \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)} \right) x^{(i)}$	$\theta = \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi\left(x^{(i)}\right) \right) \phi\left(x^{(i)}\right)$	$x \in \mathbb{R}^d$ $\phi(x) \in \mathbb{R}^p$ $\theta \in \mathbb{R}^{d,p}$

We identify the learning process from the GAUR

$$\theta = \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi\left(x^{(i)}\right) \right) \phi\left(x^{(i)}\right)$$

Let's initialize $\theta = \vec{0}$

$$\theta_f = \sum_{i=1}^{n} \beta_i \phi\left(x^{(i)}\right)$$

Where θ_f is the "final theta" and β_i is a scalar relating to a specific example, no less than the linear combination of the inputs.

$$\theta = \sum_{i=1}^{n} \beta_i \phi\left(x^{(i)}\right) + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi\left(x^{(i)}\right) \right) \phi\left(x^{(i)}\right)$$

$$\theta = \sum_{i=1}^{n} \left(\beta_i + \alpha \left[y^{(i)} - \theta^T \phi\left(x^{(i)}\right) \right] \right) \phi\left(x^{(i)}\right)$$

Where

$$\beta_{updated} = \beta_i + \alpha\left[y^{(i)} - \theta^T\phi\left(x^{(i)}\right)\right]$$

Then

$$\theta_{updated} = \sum_{i=1}^{n} \beta_{updated}\phi\left(x^{(i)}\right)$$

We want to find an general formula expressed in terms of β_i

$$\beta_i^{k+1} := \beta_i^k + \alpha\left(y^{(i)} - \theta^T\phi\left(x^{(i)}\right)\right)$$

$$\beta_i = \beta_i + \alpha\left(y^{(i)} - \left(\sum_{j=1}^{n}\beta_j\phi\left(x^{(j)}\right)\right)^T\phi\left(x^{(i)}\right)\right)$$

$$\beta_i = \beta_i + \alpha\left(y^{(i)} - \sum_{j=1}^{n}\beta_j\langle\phi\left(x^{(j)}\right), \phi\left(x^{(i)}\right)\rangle\right)$$

Hence, $\phi(x)$ could be infinite dimensional, but still representable with linear combinations.

	θ_1	θ_2	…	θ_d	θ_∞
β_1	$x_1^{(1)}$	$x_2^{(1)}$	…	$x_d^{(1)}$	$x_\infty^{(1)}$
⋮	⋮	⋮	⋮	⋮	⋮
β_n	$x_1^{(n)}$	$x_2^{(n)}$	…	$x_d^{(n)}$	$x_\infty^{(n)}$

Each example has one linear combination “weight” if each row.
We have successfully mapped ∞ into n dimensional by considering β!

The problem persists in the inner product between to ∞-dimensional vectors. Here the Kernel method’s come in clutch

$$K\left(x^{(i)}, x^{(j)}\right) = \langle\phi\left(x^{(i)}\right), \phi\left(x^{(j)}\right)\rangle$$

$$\beta_i = \beta_i + \alpha\left(y^{(i)} - \sum_{j=1}^{n} \beta_j K\left(x^{(i)}, x^{(j)}\right)\right)$$

$$\beta = \beta + \alpha(\vec{y} - K\beta)$$

Then

$$K(x, z) = \left(x^T z + c\right)^2 \iff \langle \phi(x)\phi(z) \rangle$$

K allows us to perform this calculation computationally much less expensively but in a mathematically equivalent form

We can make a prediction in the following form

$$h_\theta(x) = \beta_i \langle \phi\left(x^{(i)}\right), \phi(x) \rangle$$

Finding or proving a kernel function

In order to prove a function K is a valid kernel function, there are 2 approaches

I. Try to represent it as the inner product of a feature map (knowing the feature map itself is required)

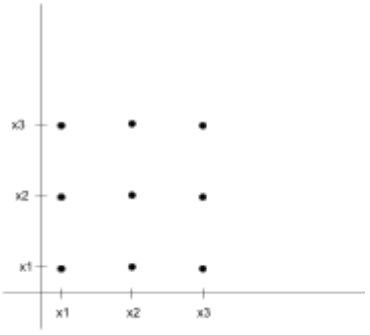
II. Take a set of examples (that have nothing to do with the dataset) $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$ and construct the kernel matrix K such that

$$K_{ij} = K\left(x^{(i)}, x^{(j)}\right)$$

And prove that $K \geq 0$ (PSD).
This is *Marcel's Theorem* and it is *necessary and sufficient* to prove the validity of a kernel function.

Let's zoom into the second method. Let $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$ be a set of examples. Then

$$K_{ij} = x^{(a,\, b)}$$



becomes

$$\begin{bmatrix} K\left(x^{(1)}, x^{(1)}\right) & K\left(x^{(1)}, x^{(2)}\right) & & \\ & \ddots & & \\ & & & K\left(x^{(3)}, x^{(3)}\right) \end{bmatrix} = \begin{bmatrix} \left(x^{(1)}\right)^T x^{(1)} & & & \\ & \ddots & & \\ & & & \left(x^{(3)}\right)^T x^{(3)} \end{bmatrix}$$