CS 229, Spring 2023
Section #3 Solutions: Naive Bayes, Kernels

---

1. **Naive Bayes (NB):** In Naive Bayes, we make a **STRONG** assumption that the data x_i are conditionally independent given labels y when modeling $p(x|y)$. For instance, if y is a binary value for classifying spam emails with $y = 1$ for spam and $y = 0$ otherwise, and x is a vector of key words that could be in the email, then e.g. $p(x_2|y) = p(x_2|y, x_1)$ (i.e. knowledge of whether the first key word appearing in the email does not lead to any implications involving the second key word). This assumption leads to

$$p(x_1, ..., x_d|y) = \prod_{j=1}^{d} p(x_j|y)$$

parameterized by

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

after maximizing the likelihood function

$$\ell = \log\left(\prod_{i=1}^{n} p(x^{(i)}, y^{(i)})\right)$$

   (a) Suppose we have 100 examples with 7 features each. How many parameters (ϕ's) would we need to fully specify a Naive Bayes model for our data?
   Answer: We need 1 for ϕ_y, 7 for $\phi_{j|y=0}$, and 7 for $\phi_{j|y=1}$. So we need 15 in total.

   (b) (Midterm Fall 2018) What is the decision boundary of a Naive Bayes classifier? Write the decision boundary in terms of ϕ's (Hint: Start with $\log p(y = 1|x) = \log p(y = 0|x)$)
   Answer: First, notice:

$$\log p(y = 1|x) = \log p(x|y = 1) + \log p(y = 1) - \log p(x)$$

   The above follows from Bayes' rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

   and properties of logarithms. Then, we can expand $\log p(x|y = 1)$ to the following:

$$\begin{aligned}\log p(x|y = 1) &= \sum_{j=1}^{d} \{x_j \log p(x_j = 1|y = 1) + (1 - x_j) \log p(x_j = 0|y = 1)\} \\ &= x^\top \log \phi_{*|y=1} + (1 - x)^\top \log(1 - \phi_{*|y=1})\end{aligned}$$

Where $\phi_{*|y=1}$ denotes the vector of all $\phi_{j|y=1}$. Recall that $\forall j, x_j \in \{0, 1\}$, so in the above equation we use each x_j as an indicator to select the correct log probability. Substituting this back into our equation for $\log p(y = 1|x)$, we have:

$$\log p(y = 1|x) = x^\top \log \phi_{*|y=1} + (1 - x)^\top \log(1 - \phi_{*|y=1}) + \log \phi_y - \log p(x)$$

We can use similar reasoning for $\log p(y = 0|x)$:

$$\log p(y = 0|x) = x^\top \log \phi_{*|y=0} + (1 - x)^\top \log(1 - \phi_{*|y=0}) + \log(1 - \phi_y) - \log p(x)$$

Where $\phi_{*|y=0}$ similarly denotes the vector of all $\phi_{j|y=0}$. Now, when we set $\log p(y = 1|x) = \log p(y = 0|x)$, the $-\log p(x)$ terms on each side cancel out, and we have:

$$\begin{aligned} &x^\top \log \phi_{*|y=1} + (1 - x)^\top \log(\vec{1} - \phi_{*|y=1}) + \log \phi_y = \\ &x^\top \log \phi_{*|y=0} + (1 - x)^\top \log(1 - \phi_{*|y=0}) + \log(1 - \phi_y) \end{aligned}$$

Bringing everything over to one side, we have:

$$\begin{aligned} &x^\top \log \frac{\phi_{*|y=1}}{\phi_{*|y=0}} + (1 - x)^\top \log \left(\frac{1 - \phi_{*|y=1}}{1 - \phi_{*|y=0}} \right) + \log \left(\frac{\phi_y}{1 - \phi_y} \right) = 0 \\ &x^\top \log \left(\frac{\phi_{*|y=1}(1 - \phi_{*|y=0})}{\phi_{*|y=0}(1 - \phi_{*|y=1})} \right) + \mathbf{1}^T \log \left(\frac{\phi_y(1 - \phi_{*|y=1})}{(1 - \phi_y)(1 - \phi_{*|y=0})} \right) = 0 \end{aligned}$$

Here, $\log$ and division are performed element-wise across vectors. This gives us a linear function of x in terms of ϕ's.

2. **Kernels:** In Problem Set 1, you had a problem involving linear regression with respect to a polynomial of a single input feature x. In the real world however, you most likely have more than 1 input feature.

   (a) Consider the case when you have 2000 examples of 1000 input features each, like $\vec{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, ..., x_{1000}^{(i)}]$. And you would like to include all polynomials of degree equal to 2:

   $$\phi(\vec{x}) = [1, x_1^2, x_2^2, x_3^2, ..., x_1x_2, x_1x_3, ...,]$$

   i. In its original form (without featurization), how many parameters do we need to fit for a basic, ordinary least squares model? (Hint: the model is $y = \theta^T\vec{x}$)
   Answer: We need 1001 parameters (one for the intercept)

   ii. In the featurized form (involving ϕ), how many parameters do we need to fit a least squares model? (Hint: the model is $y = \theta^T\phi(\vec{x})$)
   Answer: We need $1000^2 + 1 = 1e6 + 1$ parameters!

   (b) The core idea of kernels is that we represent our θ as a linear combination of featurized examples, i.e.

   $$\theta = \sum_{i=1}^{n} \beta_i \phi(\vec{x}^{(i)})$$

   Note that β_i is a scalar (i.e. a number) not a vector

   i. The original gradient update rule is

   $$\theta := \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T\vec{x}^{(i)})\vec{x}^{(i)}$$

   What is the new update rule with features?
   Answer:

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

   ii. What is the new update rule in terms of a particular β_i?
   Answer:

   $$\begin{aligned}
   \theta &:= \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T\phi(\vec{x}^{(i)}))\phi(\vec{x}^{(i)}) \\
   &:= \sum_{i=1}^{n} \beta_i\phi(\vec{x}^{(i)}) + \alpha \sum_{i=1}^{n} (y^{(i)} - \sum_{j=1}^{n} \beta_j\phi(\vec{x}^{(j)})^T\phi(\vec{x}^{(i)}))\phi(\vec{x}^{(i)}) \\
   &:= \sum_{i=1}^{n} [\beta_i + \alpha(y^{(i)} - \sum_{j=1}^{n} \beta_j\phi(\vec{x}^{(j)})^T\phi(\vec{x}^{(i)}))]\phi(\vec{x}^{(i)})
   \end{aligned}$$

   Notice above that we have represented the new θ as a linear combination of $\phi(\vec{x}^{(i)})$, thus we can say that the new β is:

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

This leads to a $\phi(\vec{x}^{(j)})^T\phi(\vec{x}^{(i)})$ term, which allows us to use the "kernel trick" and replace the dot product with an $n \times n$ "look-up" matrix K for

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

iii. How many parameters do we need to fit a kernelized least squares model?
Answer: Since we have reparametrized our model to use β's, we now have 2000 parameters (one for each featurized example), much less than the 1000001 parameters we had earlier.

iv. The kernel trick lets us use less parameters and save space when doing gradient descent, but calculating the kernel matrix may still take a extremely long time. Naively multiplying every featurized vector with each other in this case takes $O(n^2d^2)$ time (note that d can be very large). How can we compute it more efficiently? **Answer:** We can precompute $K(\vec{x}^{(j)}, \vec{x}^{(i)})$ before we start any of the gradient updates. However, how do we efficiently calculate $K(\vec{x}^{(j)}, \vec{x}^{(i)})$? Notice that our feature contains every polynomial of degree equals 2, so we can compute the kernel super efficiently by

$$\begin{aligned}
K(\vec{x}^{(j)}, \vec{x}^{(i)}) &= \phi(\vec{x}^{j})^T\phi(\vec{x}^{i}) \\
&= 1 + \sum_{k=1}^{d}\sum_{l=1}^{d} x_k^{(j)}x_l^{(j)}x_k^{(i)}x_l^{(i)} \quad \text{(the 1 comes from the intercept)} \\
&= 1 + \sum_{k=1}^{d} x_k^{(j)}x_k^{(i)} \sum_{l=1}^{d} x_l^{(j)}x_l^{(i)} \\
&= 1 + (\sum_{k=1}^{d} x_k^{(j)}x_k^{(i)})^2 \\
&= 1 + ((\vec{x}^{(j)})^T\vec{x}^{(i)})^2
\end{aligned}$$

We have reduced this to a computation that takes $O(n^2d)$ time (there are n^2 pairs of examples to compute the kernel function for, and each takes d time)!

(c) (Midterm Fall 2018) Let us attempt to kernelize some other update formulas. Consider the following formula:

$$\text{step 1:} \quad c^{(i)[t+1]} := \arg\min_j ||x^{(i)} - \mu_j^{[t]}||^2$$

$$\text{step 2:} \quad \mu_j^{[t+1]} := \frac{\sum_{i=1}^{m} 1\{c^{(i)[t+1]} = j\}x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)[t+1]} = j\}}$$

We seek to combine this into a single formula while also applying the kernel trick to allow infinite dimensional features. Complete the derivation below:

$$\begin{aligned}
c^{(i)[t+1]} &:= \arg\min_j ||\phi(x^{(i)}) - \mu_j||^2 \\
&:=
\end{aligned}$$

Answer:

$$
\begin{aligned}
\tilde{c}^{(i)} &:= \arg\min_j \|\phi(x^{(i)}) - \mu_j\|^2 \text{ (abusing notation)} \\
&= \arg\min_j \phi(x^{(i)})^T\phi(x^{(i)}) + \mu_j^T\mu_j - 2\phi(x^{(i)})^T\mu_j \\
&= \arg\min_j \phi(x^{(i)})^T\phi(x^{(i)}) - 2\phi(x^{(i)})^T\left(\frac{\sum_{l\in S_j}\phi(x^{(l)})}{|S_j|}\right) \\
&\quad + \left(\frac{\sum_{l\in S_j}\phi(x^{(l)})}{|S_j|}\right)^T\left(\frac{\sum_{l\in S_j}\phi(x^{(l)})}{|S_j|}\right) \text{ (where } S_j = \{l : c^{(l)} = j\}) \\
&= \arg\min_j \left(K(x^{(i)}, x^{(i)}) - 2\frac{1}{|S_j|}\sum_{l\in S_j} K(x^{(i)}, x^{(l)}) + \frac{1}{|S_j|^2}\sum_{(l,p)\in S_j^2} K(x^{(l)}, x^{(p)})\right) \\
&= \arg\min_j \left(-2\frac{1}{|S_j|}\sum_{l\in S_j} K(x^{(i)}, x^{(l)}) + \frac{1}{|S_j|^2}\sum_{(l,p)\in S_j^2} K(x^{(l)}, x^{(p)})\right)
\end{aligned}
$$