10-315 Intro to Machine Learning (PhD)

Lecture 5: Naive Bayes

Leila Wehbe

Carnegie Mellon University

Machine Learning Department

Reading: **Generative and Discriminative Classifiers** by Tom Mitchell.

LECTURE OUTCOMES:

- Conditional Independence
- Naïve Bayes, Gaussian Naive Bayes
- Practical Examples

Links (use the version you need)

- **Notebook**
- **PDF slides**

Assume you want to build a classifier for new customers

O Is older than 35 years	I Has Personal Income	S Is a Student	J Birthday before July 1st	Y Buys computer
0	1	0	0	0
0	1	0	1	0
1	1	0	1	1
1	1	0	1	1
1	0	1	0	1
1	0	1	1	0
0	0	1	0	1
0	1	0	0	0
0	0	1	1	1
0	1	1	0	1
0	0	1	1	1
1	1	0	1	1
0	1	1	0	1
1	1	0	0	0

What to predict for the next customer? We want to find $P(Y=0|O=0, I=0, S=1, J=1)$

O Is older than 35 years	I Has Personal Income	S Is a Student	J Birthday before July 1st	Y Buys computer
0	1	1	1	?

How many parameters must we estimate?

Suppose $X = (X_1, X_2)$ where X_i and Y are boolean random variables
How many parameters do we need to estimate to know $P(Y|X1, X2)$?

How many parameters must we estimate?

Suppose $X = (X_1, X_2)$ where X_i and Y are boolean random variables
How many parameters do we need to estimate to know $P(Y|X1, X2)$?

X_1	X_2	$P(Y = 1 \mid X_1, X_2)$	$P(Y = 0 \mid X_1, X_2)$
0	1	0.1	0.9
1	0	0.24	0.76
0	1	0.54	0.46
1	1	0.23	0.77

4 parameters: $P(Y = 0 \mid X_1, X_2) = 1 - P(Y = 1 \mid X_1, X_2)$

How many parameters must we estimate?

Suppose $X = (X_1, X_2, \ldots, X_d)$ where X_i and Y are boolean random variables
How many parameters do we need to estimate to know $P(Y|X_1, \ldots X_d)$?

How many parameters must we estimate?

Suppose $X = (X_1, X_2, \ldots, X_d)$ where X_i and Y are boolean random variables
How many parameters do we need to estimate to know $P(Y|X_1, \ldots X_d)$?

X_1	X_2	X_3	...	X_d	$P(Y=1 \mid X)$	$P(Y=0 \mid X)$
0	0	0	...	0	0.1	0.9
0	0	0	...	1	0.24	0.76
...	...	...	...	...	...	...
...	...	...	...	...	...	...
1	1	1	...	1	0.52	0.48

How many parameters must we estimate?

Suppose $X = (X_1, X_2, \ldots, X_d)$ where X_i and Y are boolean random variables
How many parameters do we need to estimate to know $P(Y|X_1, \ldots X_d)$?

X_1	X_2	X_3	...	X_d	$P(Y = 1 \mid X)$	$P(Y = 0 \mid X)$
0	0	0	...	0	0.1	0.9
0	0	0	...	1	0.24	0.76
...	...	...	...	...	...	...
...	...	...	...	...	...	...
1	1	1	...	1	0.52	0.48

2^d rows! (2^d parameters!).
If we have 30 boolean X_i's: ~ 1 billion rows!

Last Lecture we asked:

Can we just estimate P(Y|X) in this fashion and be done?

We might not have enough data. For example consider having 100 attributes of people:

- how many rows will we have? $2^{100} > 10^{30}$
- how many people on earth? $< 10^{10}$
- 99.99% of rows will not have training examples!

Can we use Bayes Rule to reduce the number of parameters?

We used bayes rule last lecture to express marginal and conditional probabilities of the data and parameter θ:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

We can also express the conditional distribution of Y given X:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

BTW, this notation is a shorthand for:

$$(\forall i, j)\ P(Y = y_i|X = x_j) = \frac{P(X = x_j|Y = y_i)P(Y = y_i)}{P(X = x_j)}$$

Equivalently:

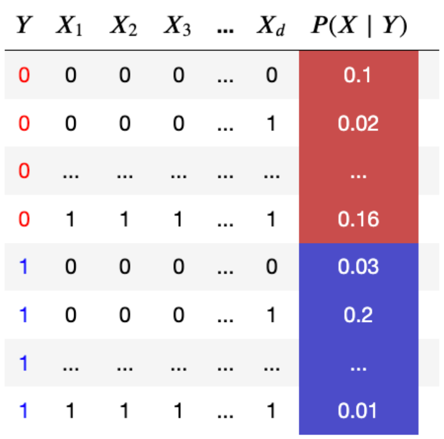
$$(\forall i, j)\ P(Y = y_i|X = x_j) = \frac{P(X = x_j|Y = y_i)P(Y = y_i)}{\sum_k P(X = x_i|Y = y_k)P(Y = y_k)}$$

Does Bayes Rule help reduce the number of parameters?

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- $P(X)$ is the same for both classes, so we might be able to avoid computing it (also, I can get it for free if I already know $P(X|Y)$ and $P(Y)$).
- $P(Y)$ is only 1 parameter.
- What about $P(X|Y)$? i.e. $P(X_1, X_2, \ldots X_d|Y)$. How many parameters do we need?

How many parameters do we need for $P(X_1, X_2, \ldots X_d|Y)$?



Y	X_1	X_2	X_3	...	X_d	$P(X \mid Y)$
0	0	0	0	...	0	0.1
0	0	0	0	...	1	0.02
0	...	...	...	...	...	...
0	1	1	1	...	1	0.16
1	0	0	0	...	0	0.03
1	0	0	0	...	1	0.2
1	...	...	...	...	...	...
1	1	1	1	...	1	0.01

- How many parameters for the red cells? $2^d - 1$
- Should I compute parameters for the blue cells as well? yes, $2^d - 1$

Does Bayes Rule help reduce the number of parameters?

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- $P(X)$ is the same for both classes, so we might be able to avoid computing it (also, I can get it for free if I already know $P(X|Y)$ and $P(Y)$).
- $P(Y)$ is only 1 parameter.
- $P(X_1, X_2, \dots X_d|Y)$: $2(2^d - 1)$ parameters

Still too many parameters!
If we have 30 boolean X_i's: ~ 2 billion!

Solution:

- 1- Be smart about how to estimate probabilities from sparse data
    - maximum likelihood estimates
    - maximum a posteriori estimates
- 2- Be smart about how to represent joint distributions
    - Bayes networks, graphical models, conditional independence

Be smart about how to represent joint distributions

Conditional Independence:

Definition: X is conditionally independent of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z.

$$(\forall i, j, k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_j)$$

- Which we often write:

$$P(X|Y, Z) = P(X|Z)$$

- For example: $P(\text{thunder}|\text{rain}, \text{lightning}) = P(\text{thunder}|\text{lightning})$
    - Thunder is independent of rain **given lightning**.
    - Once we know there if there is or there is lightning, no more information is provided by the value of rain.
    - **This does not mean that thunder is independent of rain**.

The Naïve Bayes Algorithm

- Naïve Bayes is a classifier that assumes conditional independence of the variables X_i given the label Y.
  - For example: $P(X_1|X_2, Y) = P(X_1|Y)$
- How does this assumption simplify $P(X_1, X_2|Y)$?

$$\begin{aligned} P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) && \text{(chain rule)} && (1) \\ &= P(X_1|Y)P(X_2|Y) && \text{(conditional independence)} && (2) \end{aligned}$$

- In general:

$$P(X_1, \dots, X_d|Y) = \prod_i P(X_i|Y) \tag{3}$$

- How many parameters do we need to describe $P(X_1 \dots X_n|Y)$?
  - Without the conditional independence assumption: $2(2^d - 1)$
  - With the conditional independence assumption:

The Naïve Bayes Algorithm

- Naïve Bayes is a classifier that assumes conditional independence of the variables X_i given the label Y.
  - For example: $P(X_1|X_2, Y) = P(X_1|Y)$
- How does this assumption simplify $P(X_1, X_2|Y)$?

$$\begin{aligned} P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \quad \text{(chain rule)} \qquad (4) \\ &= P(X_1|Y)P(X_2|Y) \quad \text{(conditional independence)} \qquad (5) \end{aligned}$$

- In general:

$$P(X_1, \ldots, X_d|Y) = \prod_i P(X_i|Y) \qquad (6)$$

- How many parameters do we need to describe $P(X_1 \ldots X_n|Y)$?
  - Without the conditional independence assumption: $2(2^d - 1)$
  - With the conditional independence assumption: $2d$

The Naïve Bayes Algorithm

- Naïve Bayes assumes conditional independence of the X_i's:

$$P(X_1, \ldots, X_d | Y) = \prod_i P(X_i | Y)$$

(more on this assumption soon!)

- Using Bayes rule with that assumption:

$$P(Y = y_k | X_1, \ldots, X_d) = \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{P(X)}$$

- Train the algorithm
  - estimate $P(X_i | Y = y_k)$ and $P(Y = y_k)$

- To classify, pick the most probable Y^{new} for a new sample $X^{\text{new}} = (X_1^{\text{new}}, X_2^{\text{new}}, \ldots, X_d^{\text{new}})$ as:

$$Y^{\text{new}} \leftarrow \underset{y_k}{\text{argmax}}\, P(Y = y_k) \prod_i P(X_i^{\text{new}} | Y = y_k)$$

Naïve Bayes - Training and Prediction Phase - Discrete X_i

Training:

- Estimate $\pi_k \equiv P(Y = y_k)$, get $\hat{\pi}_k$
- Estimate $\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k)$, get $\hat{\theta}_{ijk}$
    - θ_{ijk} is estimated for each label y_k:
        - For each variable X_i:
            - For each value x_{ij} that X_i can take.
    - Example: if X_1 is binary, $P(X_1 | Y = 0)$ is a bernouilli distribution, where:
        - the probability of $X_1 = 0$ given $Y = 0$ is θ_{100}
        - the probability of $X_1 = 1$ given $Y = 0$ is θ_{110}
        - $\theta_{100} = 1 - \theta_{110}$.

Prediction: Classify Y^{new}

$$
\begin{aligned}
Y^{\text{new}} &= \underset{y_k}{\operatorname{argmax}}\, P(Y = y_k) \prod_i P(X_i^{\text{new}} = x_j^{\text{new}} | Y = y_k) && (7) \\
&= \underset{y_k}{\operatorname{argmax}}\, \hat{\pi}_k \prod_i \hat{\theta}_{i, X_i^{\text{new}}, k} && (8)
\end{aligned}
$$

But... how do we estimate these parameters?

Naïve Bayes - Training Phase - Discrete X_i - Maximum (Conditional) Likelihood Estimation

- $P(X|Y=y_k)$ has parameters θ_{ijk}, one for each value x_{ij} of each X_i. $P(Y)$ has parameters π_k.
- To follow the MLE principle, we pick the parameters π and θ that maximizes the (**conditional**) likelihood of the data given the parameters.

- To estimate:
  - Compute

$$\hat{\pi}_k = \hat{P}(Y=y_k) = \frac{\#D(Y=y_k)}{|D|}$$

  - For each label y_k:
    - For each variable X_i:
      - For each value x_{ij} that X_i can take, compute:

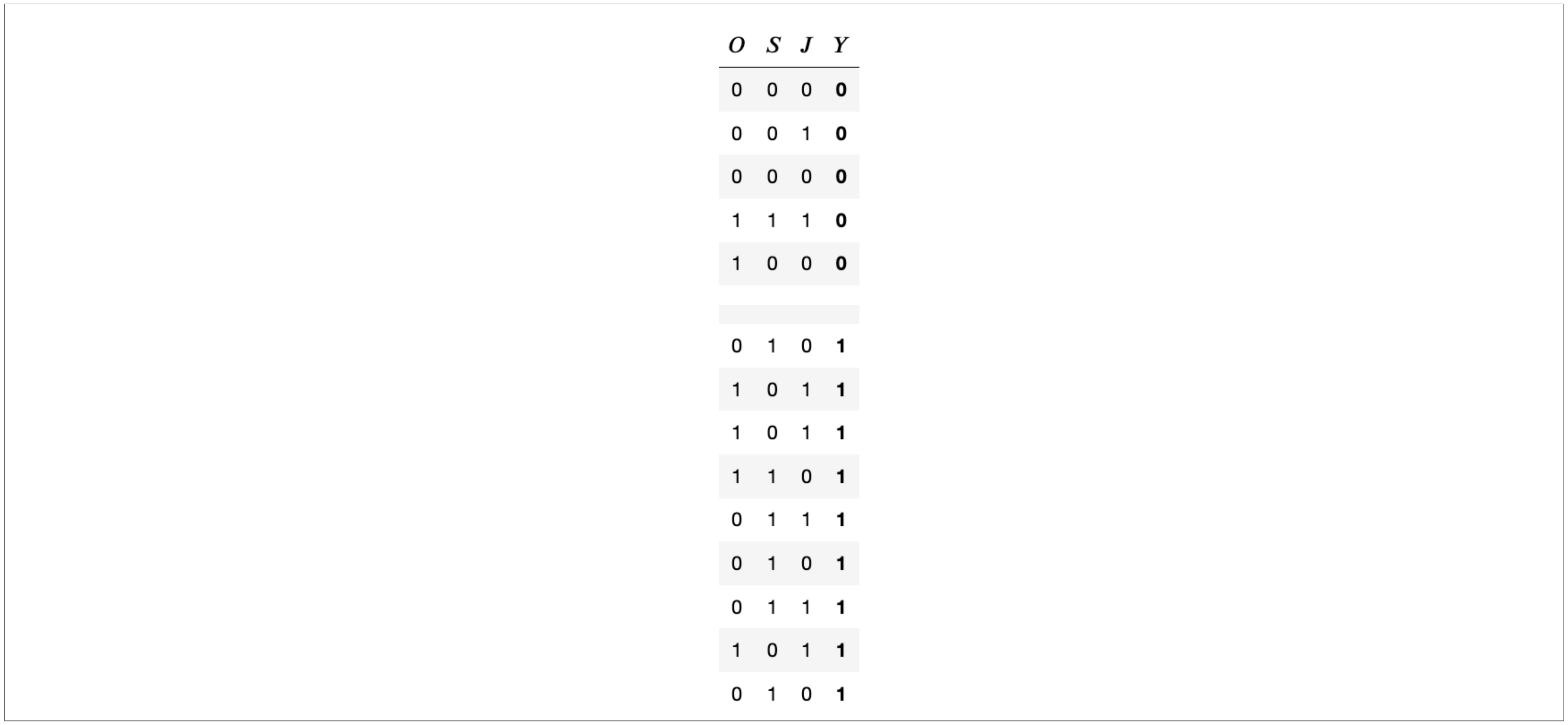
$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij}|Y=y_k) = \frac{\#D(X_i = x_{ij} \wedge Y = y_k)}{\#D(Y=y_k)}.$$

Let's train!

O **Is older than 35 years**	I **Has Personal Income**	S **Is a Student**	J **Birthday before July 1st**	Y **Buys computer**
0	1	0	0	0
0	1	0	1	0
1	1	0	1	1
1	1	0	1	1
1	0	1	0	1
1	0	1	1	0
0	0	1	0	1
0	1	0	0	0
0	0	1	1	1
0	1	1	0	1
0	0	1	1	1
1	1	0	1	1
0	1	1	0	1
1	1	0	0	0

Let’s train!

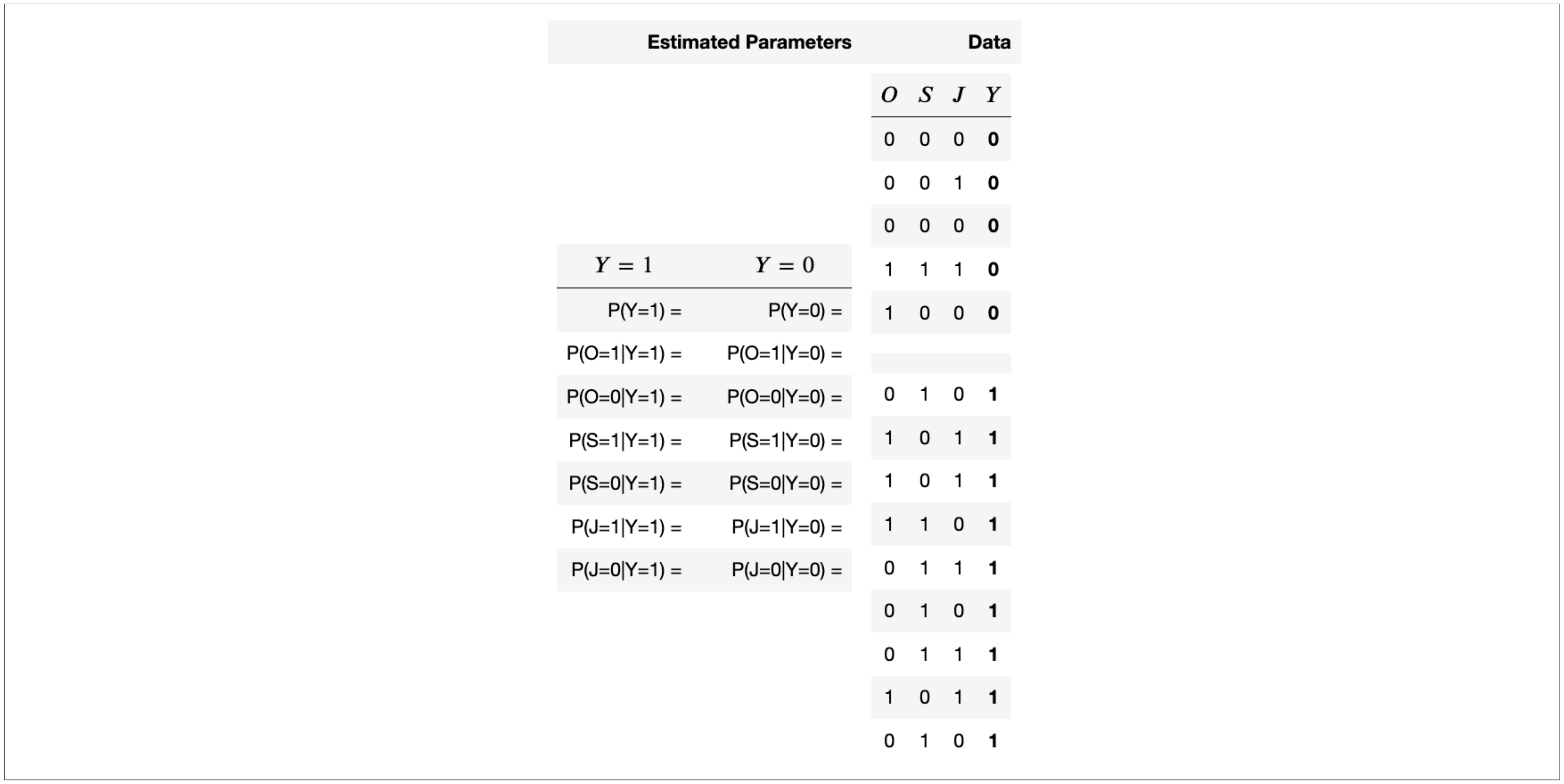
Removed one variable to simplify the problem + changed order of samples.
O= Is older than 35, S= Is a student, J = Birthday before July 1, Y= buys computer



O	S	J	Y
0	0	0	**0**
0	0	1	**0**
0	0	0	**0**
1	1	1	**0**
1	0	0	**0**
0	1	0	**1**
1	0	1	**1**
1	0	1	**1**
1	1	0	**1**
0	1	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	1	**1**
0	1	0	**1**

Let’s train!

O= Is older than 35, S= Is a student, J = Birthday before July 1, Y= buys computer



Estimated Parameters

$Y = 1$	$Y = 0$
P(Y=1) =	P(Y=0) =
P(O=1|Y=1) =	P(O=1|Y=0) =
P(O=0|Y=1) =	P(O=0|Y=0) =
P(S=1|Y=1) =	P(S=1|Y=0) =
P(S=0|Y=1) =	P(S=0|Y=0) =
P(J=1|Y=1) =	P(J=1|Y=0) =
P(J=0|Y=1) =	P(J=0|Y=0) =

Data

O	S	J	Y
0	0	0	**0**
0	0	1	**0**
0	0	0	**0**
1	1	1	**0**
1	0	0	**0**
0	1	0	**1**
1	0	1	**1**
1	0	1	**1**
1	1	0	**1**
0	1	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	1	**1**
0	1	0	**1**

Let’s train!

O= Is older than 35, S= Is a student, J = Birthday before July 1, Y= buys computer

Estimated Parameters

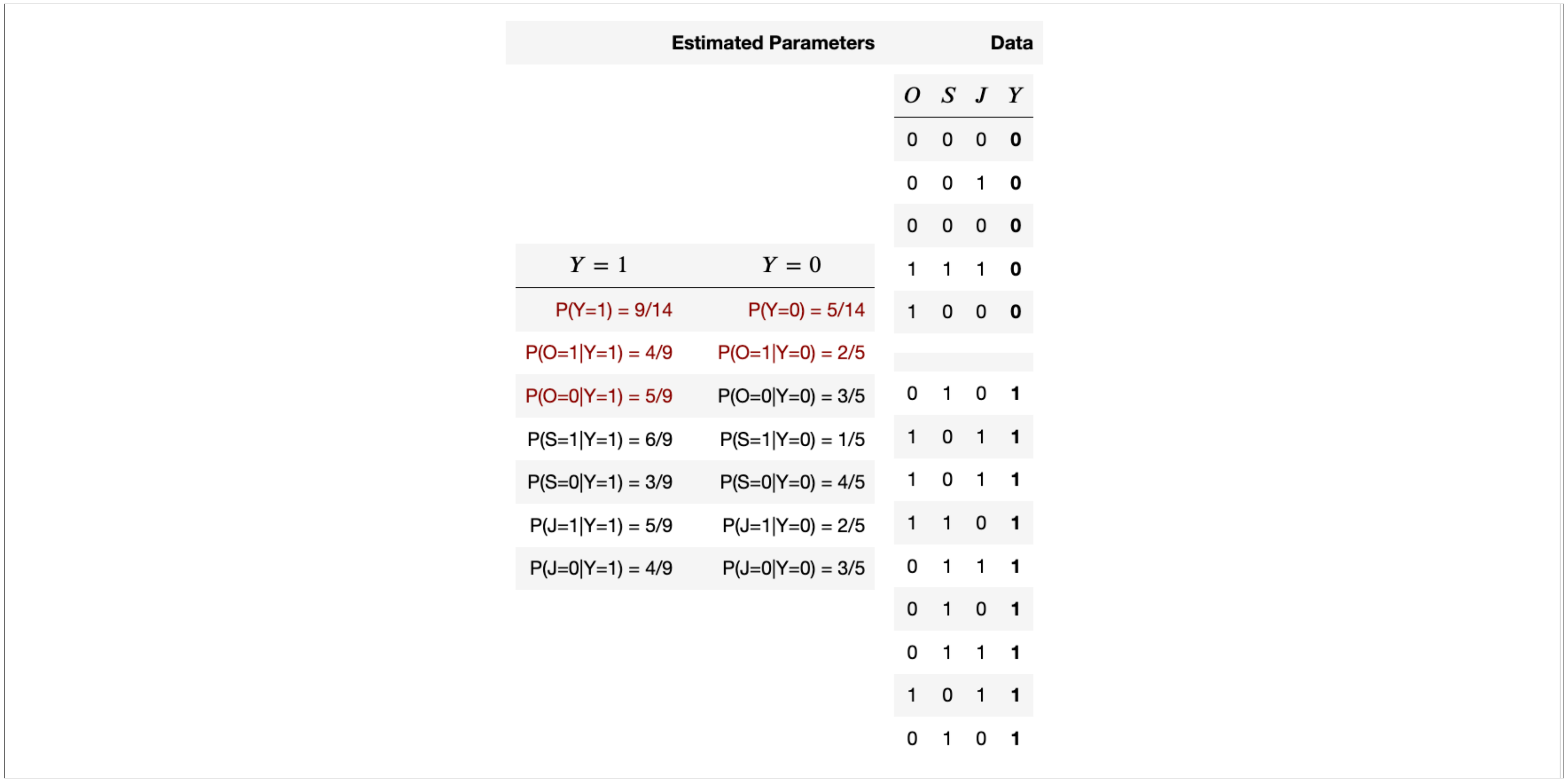
$Y = 1$	$Y = 0$
P(Y=1) = 9/14	P(Y=0) = 5/14
P(O=1|Y=1) =	P(O=1|Y=0) =
P(O=0|Y=1) =	P(O=0|Y=0) =
P(S=1|Y=1) =	P(S=1|Y=0) =
P(S=0|Y=1) =	P(S=0|Y=0) =
P(J=1|Y=1) =	P(J=1|Y=0) =
P(J=0|Y=1) =	P(J=0|Y=0) =

Data

O	S	J	Y
0	0	0	**0**
0	0	1	**0**
0	0	0	**0**
1	1	1	**0**
1	0	0	**0**
0	1	0	**1**
1	0	1	**1**
1	0	1	**1**
1	1	0	**1**
0	1	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	1	**1**
0	1	0	**1**

Let’s train!

O= Is older than 35, S= Is a student, J = Birthday before July 1, Y= buys computer



Estimated Parameters

$Y = 1$	$Y = 0$
P(Y=1) = 9/14	P(Y=0) = 5/14
P(O=1|Y=1) = 4/9	P(O=1|Y=0) = 2/5
P(O=0|Y=1) = 5/9	P(O=0|Y=0) = 3/5
P(S=1|Y=1) = 6/9	P(S=1|Y=0) = 1/5
P(S=0|Y=1) = 3/9	P(S=0|Y=0) = 4/5
P(J=1|Y=1) = 5/9	P(J=1|Y=0) = 2/5
P(J=0|Y=1) = 4/9	P(J=0|Y=0) = 3/5

Data

O	S	J	Y
0	0	0	**0**
0	0	1	**0**
0	0	0	**0**
1	1	1	**0**
1	0	0	**0**
0	1	0	**1**
1	0	1	**1**
1	0	1	**1**
1	1	0	**1**
0	1	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	1	**1**
0	1	0	**1**

Let's predict!

O= Is older than 35, S= Is a student, J = Birthday before July 1, Y= buys computer

Estimated Parameters **New Data**

$Y = 1$	$Y = 0$
P(Y=1) = 9/14	P(Y=0) = 5/14
P(O=1|Y=1) = 4/9	P(O=1|Y=0) = 2/5
P(O=0|Y=1) = 5/9	P(O=0|Y=0) = 3/5
P(S=1|Y=1) = 6/9	P(S=1|Y=0) = 1/5
P(S=0|Y=1) = 3/9	P(S=0|Y=0) = 4/5
P(J=1|Y=1) = 5/9	P(J=1|Y=0) = 2/5
P(J=0|Y=1) = 4/9	P(J=0|Y=0) = 3/5

O	S	J	Y
0	1	1	?

Let's predict!

O= Is older than 35, S= Is a student, J = Birthday before July 1, Y= buys computer

Estimated Parameters **New Data**

$Y = 1$	$Y = 0$
P(Y=1) = 9/14	P(Y=0) = 5/14
P(O=1|Y=1) = 4/9	P(O=1|Y=0) = 2/5
P(O=0|Y=1) = 5/9	P(O=0|Y=0) = 3/5
P(S=1|Y=1) = 6/9	P(S=1|Y=0) = 1/5
P(S=0|Y=1) = 3/9	P(S=0|Y=0) = 4/5
P(J=1|Y=1) = 5/9	P(J=1|Y=0) = 2/5
P(J=0|Y=1) = 4/9	P(J=0|Y=0) = 3/5

O	S	J	Y
0	1	1	?

$$Y^{\text{new}} = \underset{y_k}{\operatorname{argmax}}\, P(Y=y_k)P(O=0, S=1, J=1|Y=y_k) \tag{9}$$

$$= \underset{y_k}{\operatorname{argmax}}\, P(Y=y_k)P(O=0|Y=y_k)P(S=1|Y=y_k)P(J=1|Y=y_k) \tag{10}$$

P(Y=1) P(O=0| Y=1)P(S = 1 | Y=1) P(J = 1 | Y=1) = 9/14 * 5/9 * 6/9 * 5/9 =0.132

P(Y=0) P(O=0| Y=0)P(S = 1 | Y=0) P(J = 1 | Y=0) = 5/14 * 3/5 * 1/5 * 2/5 = 0.017

Pick label 1!

Can also compute P(Y|X):

Not required to make predictions, but we have everything we need:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

$$P(X) = \sum_k P(X|Y = y_k)P(Y = y_k)$$

P(Y=1) P(O=0| Y=1)P(S = 1 | Y=1) P(J = 1 | Y=1) = 0.132
P(Y=0) P(O=0| Y=0)P(S = 1 | Y=0) P(J = 1 | Y=0) = 0.017
P(Y=1 | O=0, S = 1, J = 1) = 0.886
P(Y=0 | O=0, S = 1, J = 1) = 0.114

Classification Accuracy

- To get an estimate of generalization performance, compute accuracy on held-out set (more about this soon).
    - Never train on your test data!
- Assume you train and use this algorithm to predict "Buy Computer?" with a large dataset, and obtain binary classification accuracy of 75%.
    - What does this mean? Is it good or bad?

Classification Accuracy

- What if 70% is the probability of $Y = 1.$ Is 75% impressive?
    - What is an easy way to obtain 70% classification accuracy?
- What is chance performance?
- What is the accuracy if I flip an unbiased coin?
- If P(Y=1)>0.5, then we can just predict 1 all the time!
    - What will be the accuracy?
- What happens if you predict Y=1 with probability 0.7 ==> this is called probability matching in cognitive science

Naïve Bayes observation 1

- Usually the X_i are not conditionally independent:

$$P(X_1, \ldots, X_d | Y) \neq \prod_i P(X_i|, Y)$$

- Even if the "naïve" conditional independence assumption is not true in the data, Naïve Bayes might still perform well and is used anyways
    - often the right classification, even when not the right probability (see [Domingos&Pazzani, 1996])

- To see the effect of the violation of this assumption, consider the extreme case in which X_i is a copy of X_k. What is the effect on $P(Y|X)$?

- To see the effect of the violation of the conditional independence assumption, consider the extreme case in which X_i is a copy of X_k. What is the effect on $P(Y|X)$?

$$\frac{P(Y=1)\ P(O=0|Y=1)\ P(S=1|Y=1)\ P(J=1|Y=1)}{P(Y=1)\ P(O=0|Y=1)\ P(S=1|Y=1)\ P(J=1|Y=1) + P(Y=0)\ P(O=0|Y=0)\ P(S=1|Y=0)\ P(J=1|Y=0)}$$

- To see the effect of the violation of the conditional independence assumption, consider the extreme case in which X_i is a copy of X_k. What is the effect on $P(Y|X)$?

$$\frac{P(Y=1)\,P(O=0|Y=1)\,P(O'=0|Y=1)\,P(S=1|Y=1)\,P(J=1|Y=1)}{P(Y=1)\,P(O=0|Y=1)\,P(O'=0|Y=1)\,P(S=1|Y=1)\,P(J=1|Y=1) + \ldots \\ \ldots\ P(Y=0)\,P(O=0|Y=0)\,P(O'=0|Y=0)\,P(S=1|Y=0)\,P(J=1|Y=0)}$$

Consider for example that $P(O=1|Y=1) > P(O=1|Y=0)$, how does $P(Y=1|O=1, O'=1, S=1, J=1)$ with the duplicated variable compare to respect $P(Y=1|O=1, S=1, J=1)$?

Naïve Bayes observation 2

What if we have an irrelevant variable?

$$\frac{P(Y=1)\,P(O=0|Y=1)\,P(S=1|Y=1)\,P(J=1|Y=1)}{P(Y=1)\,P(O=0|Y=1)\,P(S=1|Y=1)\,P(J=1|Y=1) + P(Y=0)\,P(O=0|Y=0)\,P(S=1|Y=0)\,P(J=1|Y=0)}$$

Assume J is independent of Y: $P(J|Y=0) = P(J|Y=1) = P(J)$

Does it hurt classification?

- If we have the correct estimates, then performance is not affected.
- If we have noisy estimates, performance is affected.

Naïve Bayes observation 3

- Another way to view Naïve Bayes with Boolean Y and X_is is:
- Decision rule: is this quantity greater or less than 1?

$$\frac{P(Y=1|X_1 \ldots X_d)}{P(Y=0|X_1 \ldots X_d)} = \frac{P(Y=1)P(X_1 \ldots X_d|Y=1)}{P(Y=0)P(X_1 \ldots X_d|Y=0)} \quad > \text{ or } < 1? \tag{11}$$

- Practical concern: What happens when d is large?

Naïve Bayes observation 3

- Another way to view Naïve Bayes with Boolean Y and X_is is:
- Decision rule: is this quantity greater or less than 1?

$$\frac{P(Y=1|X_1\dots X_d)}{P(Y=0|X_1\dots X_d)} = \frac{P(Y=1)P(X_1\dots X_d|Y=1)}{P(Y=0)P(X_1\dots X_d|Y=0)} \quad > \text{ or } < 1? \tag{12}$$

- Taking the log of this ratio prevents **underflow** and expresses the decision rule in a useful way (we will see later more reasons why it's useful).

$$\ln\frac{P(Y=1|X_1\dots X_d)}{P(Y=0|X_1\dots X_d)} = \ln\frac{P(Y=1)}{P(Y=0)} + \sum_i \ln\frac{P(X_i|Y=1)}{P(X_i|Y=0)} \quad > \text{ or } < 0? \tag{13}$$

- Since X_is are boolean, we can simplify the notation:
  - $\theta_{ik} = \hat{P}(X_i = 1|Y=k)$ and $1-\theta_{ik} = \hat{P}(X_i=0|Y=k)$

$$\begin{aligned}\ln\frac{P(Y=1|X_1\dots X_d)}{P(Y=0|X_1\dots X_d)} &= \ln\frac{P(Y=1)}{P(Y=0)} + \sum_i \ln\frac{P(X_i|Y=1)}{P(X_i|Y=0)} \qquad (14)\\ &= \ln\frac{P(Y=1)}{P(Y=0)} + \sum_i X_i \ln\frac{\theta_{i1}}{\theta_{i0}} + \sum_i (1-X_i)\ln\frac{1-\theta_{i1}}{1-\theta_{i0}} \qquad (15)\end{aligned}$$

Naïve Bayes observation 4

- If unlucky, our MLE estimate for $P(X_i|Y=y_k)$ might be zero.
  - for example, X_i = birthdate. xi = Jan_25_1992.

- Why worry about just one parameter out of many?

$$P(Y=y_k)P(X_1|Y=y_k)P(X_2|Y=y_k)\ldots P(X_d|Y=y_k)$$

- What happens if one of the $P(X_i|Y=y_k)$ is zero?
  - What can be done to address this?

Naïve Bayes - Training Phase - Discrete X_i

Method 1: Maximum (Conditional) Likelihood Estimation

- $P(X|Y=y_k)$ has parameters θ_{ijk}, one for each value x_{ij} of each X_i.
- To follow the MLE principle, we pick the parameters θ that maximizes the **conditional** likelihood of the data given the parameters.

Method 2: Maximum A Posteriori Probability Estimation

- To follow the MAP principle, pick the parameters θ with maximum posterior probability given the conditional likelihood of the data and the prior on θ.

Naïve Bayes - Training Phase - Discrete X_i

Method 1: Maximum (Conditional) Likelihood Estimation

To estimate:

- Compute

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D(Y = y_k)}{|D|}$$

- For each label y_k:
    - For each variable X_i:
        - For each value x_{ij} that X_i can take, compute:

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D(X_i = x_{ij} \wedge Y = y_k)}{\#D(Y = y_k)}.$$

METHOD 2: MAXIMUM A POSTERIORI PROBABILITY ESTIMATION (BETA OR DIRICHLET PRIORS)

- K: the number of values Y can take
- J: the number of values X can take (we assume here that all X_i have the same number of possible values, but this can be changed)
- Example prior for π_k where K > 2:
  - $\mathrm{Dirichlet}(\beta_\pi, \beta_\pi, .., \beta_\pi)$ prior. (optionally, you can choose different values for each parameter to encode a different weighting).
  - if $K = 2$ this becomes a Beta prior.
- Example prior for θ_{ijk} where J>2 :
  - $\mathrm{Dirichlet}(\beta_\theta, \beta_\theta, .., \beta_\theta)$ prior. (optionally, you can choose different values for each parameter to encode a different weighting, you can choose a different prior per X_i or even per label y_k).
  - if $J = 2$ this becomes a Beta prior.

Method 2: Maximum A Posteriori Probability Estimation (Beta or Dirichlet priors)

- K: the number of values Y can take
- J: the number of values X can take

These priors will act as imaginary examples that smooth the estimated distributions and prevent zero values.
To estimate:

- Compute

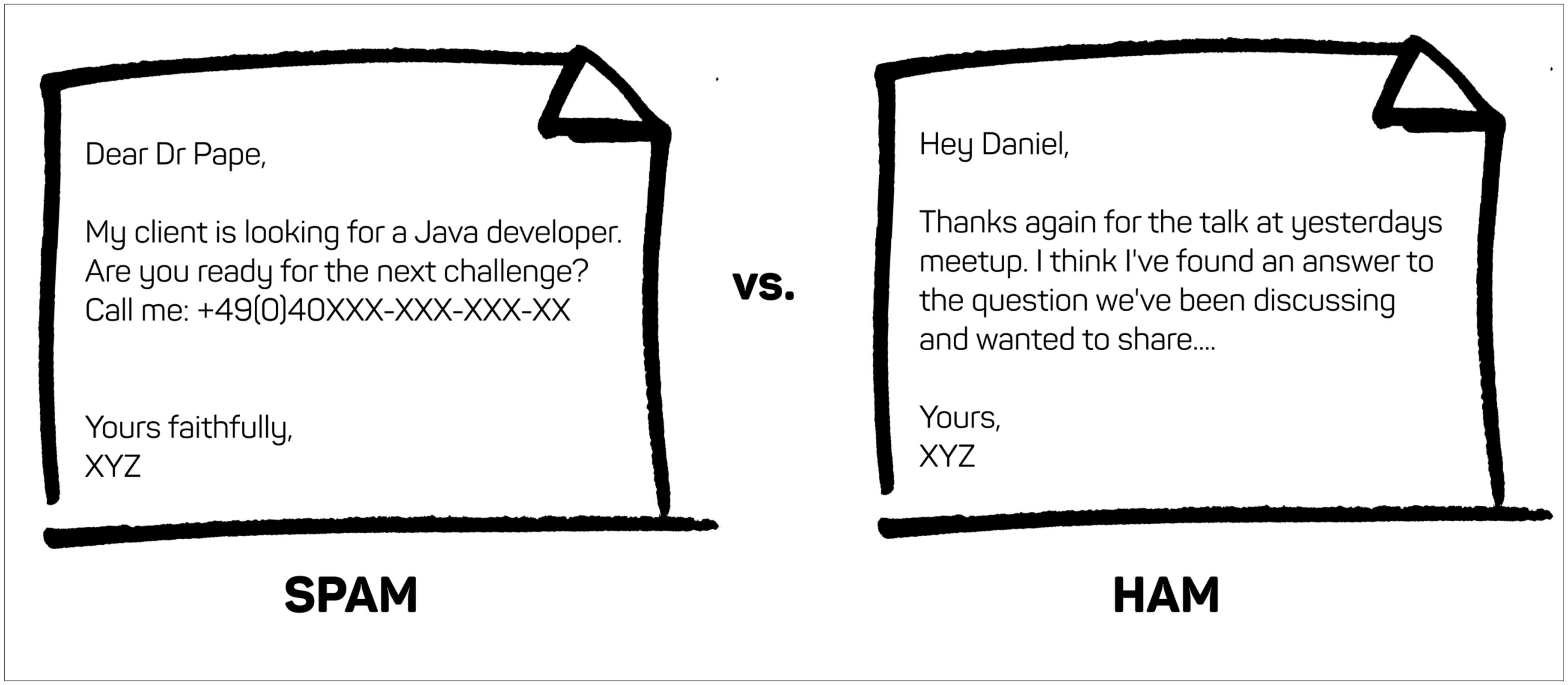
$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D(Y = y_k) + (\beta_\pi - 1)}{|D| + K(\beta_\pi - 1)}$$

- For each label y_k:
  - For each variable X_i:
    - For each value x_{ij} that X_i can take, compute:

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D(X_i = x_{ij} \wedge Y = y_k) + (\beta_\theta - 1)}{\#D(Y = y_k) + J(\beta_\theta - 1)}.$$

Example: Text classification

- Classify which emails are spam?



Dear Dr Pape,

My client is looking for a Java developer.
Are you ready for the next challenge?
Call me: +49(0)40XXX-XXX-XXX-XX

Yours faithfully,
XYZ

vs.

Hey Daniel,

Thanks again for the talk at yesterdays meetup. I think I've found an answer to the question we've been discussing and wanted to share....

Yours,
XYZ

image by Daniel Pape

- Classify which emails promise an attachment?
- Classify which web pages are student home pages?

How shall we represent text documents for Naïve Bayes?

How can we express X?

- Y discrete valued. e.g., Spam or not
- X = ?

How can we express X?

- Y discrete valued. e.g., Spam or not
- $X = (X_1, X_2, \ldots X_d)$ with d the number of words in English.

What are the limitations with this representation?

How can we express X?

- Y discrete valued. e.g., Spam or not
- $X = (X_1, X_2, \ldots X_d)$ with d the number of words in English.

What are the limitations with this representation?

- Some words always present
- Some words very infrequent
- Doesn't count how often a word appears
- Conditional independence assumption is false...

Alternative Featurization

- Y discrete valued. e.g., Spam or not
- $X = (X_1, X_2, \dots X_d)$ = document
- X_i is a random variable describing the word at position i in the document
- possible values for X_i : any word w_k in English
- X_i represents the ith word position in document
- X_1 = “I”, X_2 = “am”, X_3 = “pleased”

How many parameters do we need to estimate $P(X|Y)$? (say 1000 words per document, 10000 words in english)

Alternative Featurization

- Y discrete valued. e.g., Spam or not
- $X = (X_1, X_2, \ldots X_d)$ = document
- X_i is a random variable describing the word at position i in the document
- possible values for X_i : any word w_k in English
- X_i represents the ith word position in document
- X_1 = “I”, X_2 = “am”, X_3 = “pleased”

How many parameters do we need to estimate $P(X|Y)$? (say 1000 words per document, 10000 words in english)

Conditional Independence Assumption very useful:

- reduce problem to only computing $P(X_i|Y)$ for every X_i.

“Bag of Words” model

- Additional assumption: position doesn't matter!
  - (this is not true of language, but can be a useful assumption for building a classifier)
- assume the Xi are IID:

$$P(X_i|Y) = P(X_j|Y)(\forall i, j)$$

- we call this "Bag of Words"

Art installation in Gates building (now removed)

"Bag of Words" model

- Additional assumption: position doesn't matter!
    - (this is not true of language, but can be a useful assumption for building a classifier)
- assume the Xi are IID:

$$P(X_i|Y) = P(X_j|Y)(\forall i, j)$$

- we call this "Bag of Words"
- **Since all X_is have the same distribution, we only have to estimate one parameter per word, per class.**
- $P(X|Y=y_k)$ **is a multinomial distribution:**

$$P(X|Y=y_k) \propto \theta_{1k}^{\alpha_{1k}} \theta_{2k}^{\alpha_{2k}} \dots \theta_{dk}^{\alpha_{dk}}$$

Review of distributions

$$P(X_i = w_j) \begin{cases} \theta_1, & \text{if } X_i = w_1 \\ \theta_2, & \text{if } X_i = w_2 \\ \dots \\ \theta_k, & \text{if } X_i = w_K \end{cases} \tag{16}$$

Probability of observing a document with α_1 count of w_1, α_2 count of w_2 ... is a multinomial:

$$\frac{|D|!}{\alpha_1! \cdots \alpha_J!} \theta_1^{\alpha_1} \theta_2^{\alpha_2} \theta_3^{\alpha_3} \cdots \theta_J^{\alpha_J}$$

Review of distributions

Dirichlet Prior examples:

- if constant across classes and words:

$$P(\theta) = \frac{\theta^{\beta_\theta}\theta^{\beta_\theta}, \dots \theta^{\beta_\theta}}{\mathrm{Beta}(\beta_\theta, \beta_\theta, \dots, \beta_\theta)}$$

- if constant across classes but different for different words:

$$P(\theta) = \frac{\theta^{\beta_1}\theta^{\beta_2}, \dots \theta^{\beta_J}}{\mathrm{Beta}(\beta_1, \beta_2, \dots, \beta_J)}$$

- if different for different classes k and words:

$$P(\theta_k) = \frac{\theta^{\beta_{1k}}\theta^{\beta_{2k}}, \dots \theta^{\beta_{Jk}}}{\mathrm{Beta}(\beta_{1k}, \beta_{2k}, \dots, \beta_{Jk})}$$

MAP estimates for Bag of words:

(Dirichlet is the conjugate prior for a multinomial likelihood function)

$$\theta_{jk} = \frac{\alpha_{jk} + \beta_{jk} - 1}{\sum_m (\alpha_{mk} + \beta_{mk} - 1)} \tag{17}$$

Again the prior acts like halucinated examples
What βs should we choose?

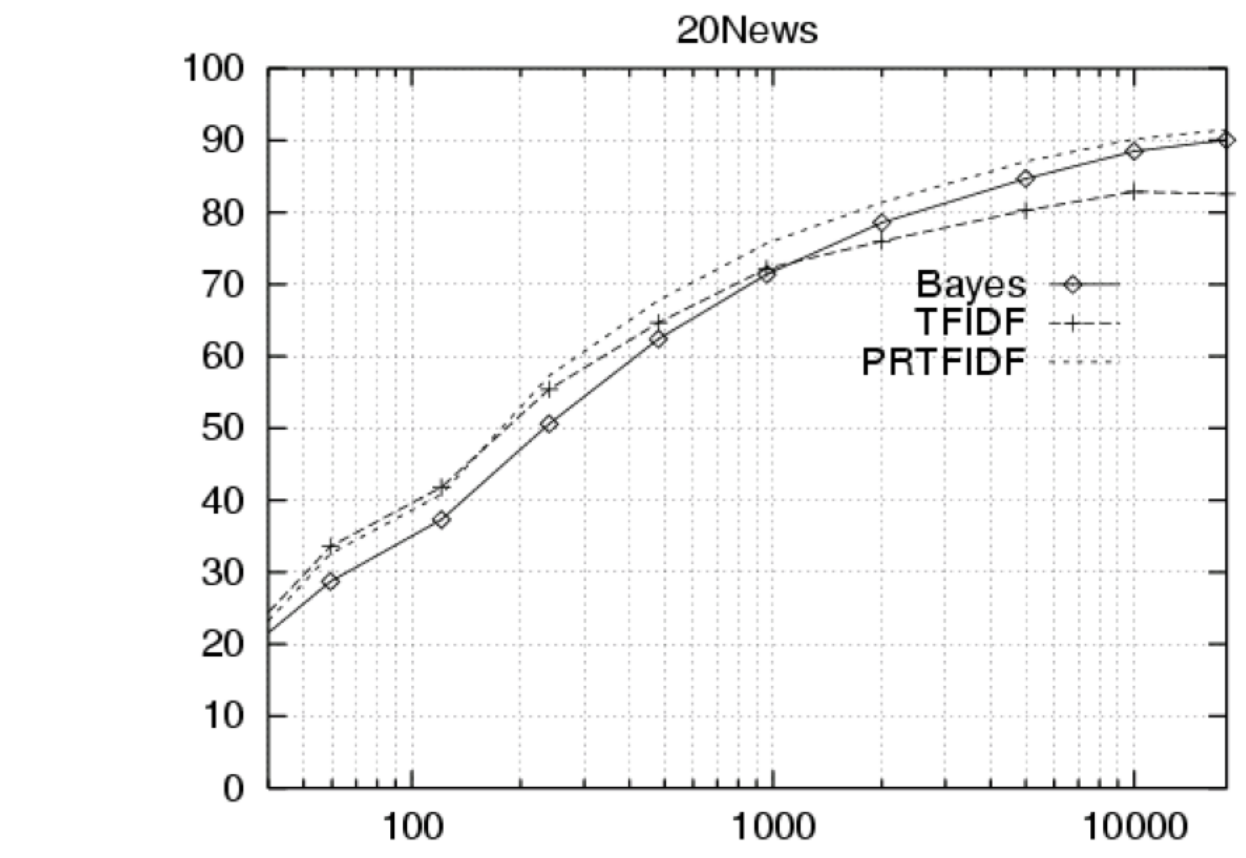
Example: Twenty NewsGroups

For code and data, see www.cs.cmu.edu/~tom/mlbook.html click on "Software and Data".
Can group labels into groups that share priors:

- comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.max.hardware, comp.windows.x
- misc.forsale
- rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey
- alt.atheism,
- soc.religion.christian,
- talk.religion.misc, talk.politics.mideast, talk.politics.misc, talk.politics.guns,
- sci.space, sci.crypt, sci.electronics, sci.med

• Naïve Bayes: 89% classification accuracy

Learning curve for 20 Newsgroups



Accuracy vs. Training set size (1/3 withheld for test)

Even if incorrect assumption, performance can be very good

Even when taking half of the email

- Assumption doesn’t hurt the particular problem?
- Redundancy?
- Leads less examples to train? Converges faster to asymptotic performance? (Ng and Jordan)

Even if incorrect assumption, performance can be very good

Even when taking half of the email

- Assumption doesn't hurt the particular problem?
- Redundancy?
- Leads less examples to train? Converges faster to asymptotic performance? (Ng and Jordan)

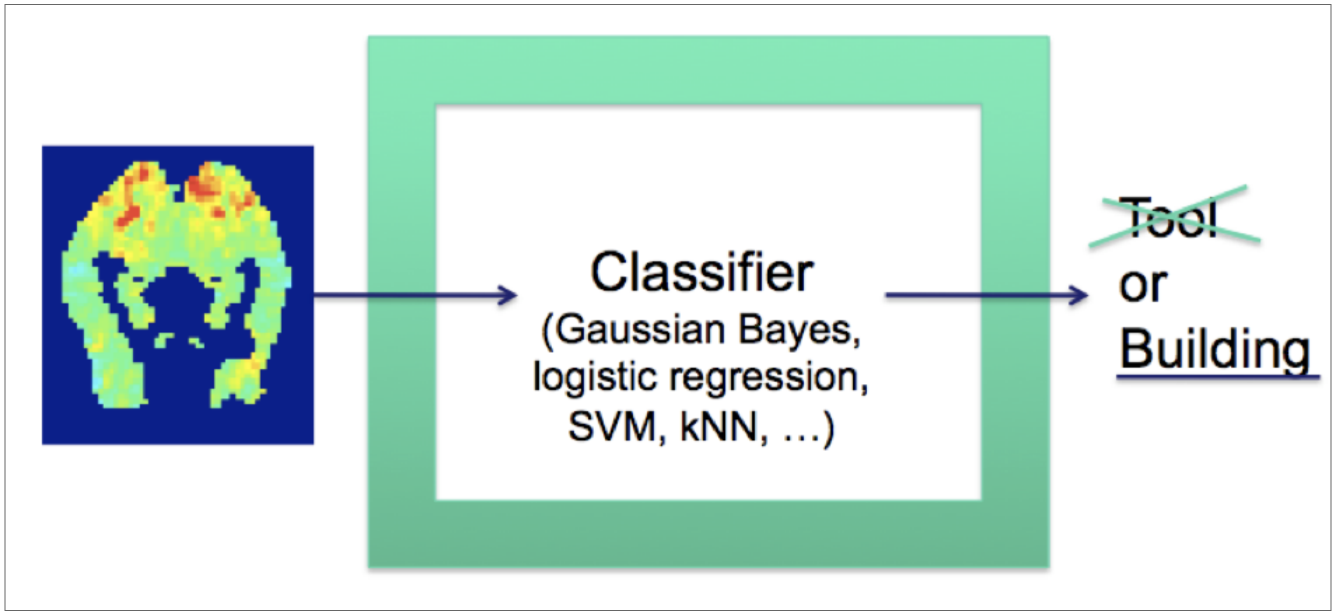
More recently, algorithms such as Transformers have become very good

- are able to capture the sequential aspect of language and produce more complex representations.
- They do have many parameters, but nowhere as much as we mentioned before (10000^{1000}).

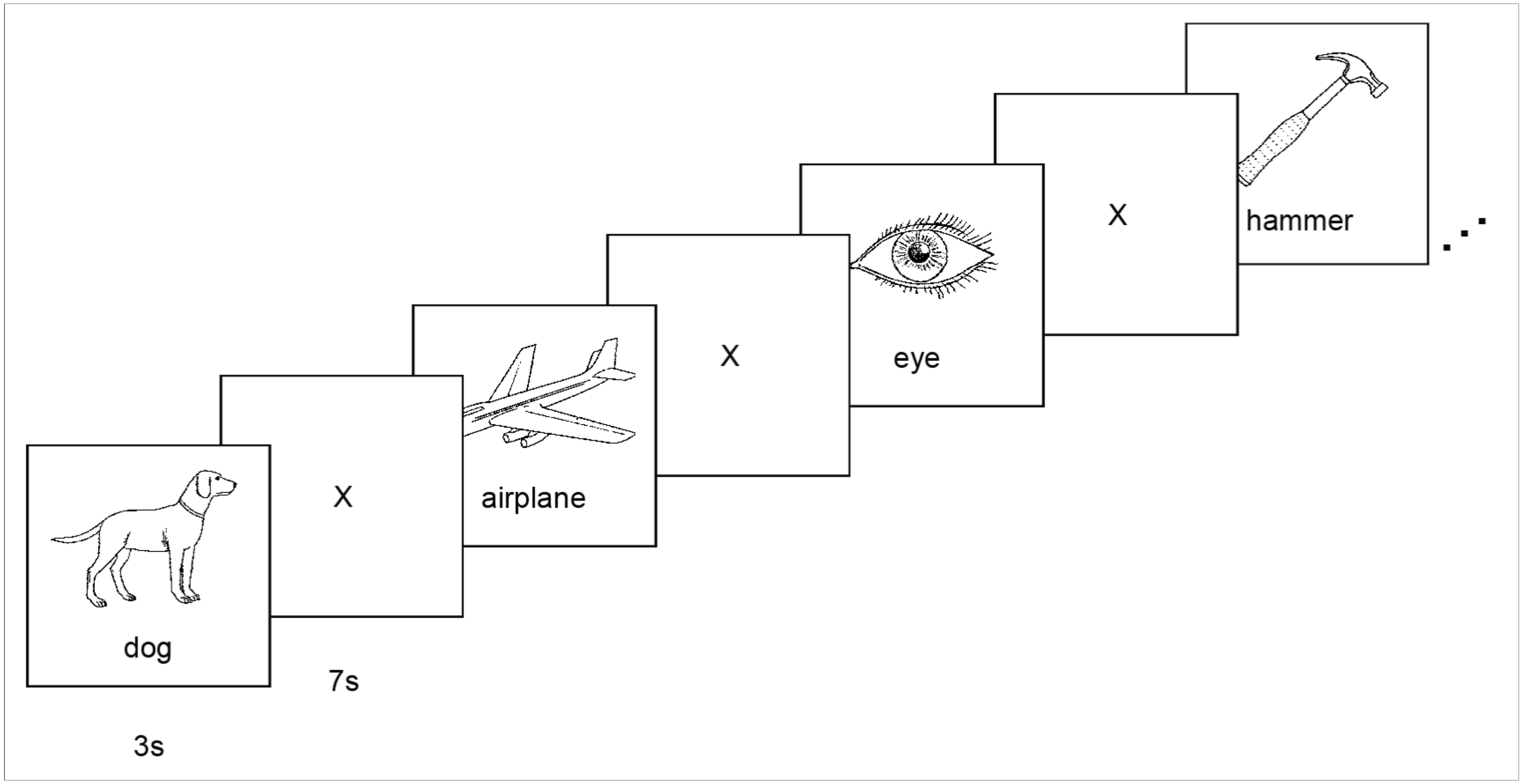
Continuous x_is

What can we do?
E.g. image classification, where X_i is real valued
Classify a person's cognitive state, based on brain image

- reading a sentence or viewing a picture?
- reading the word describing a "Tool" or "Building"?
- answering the question, or getting confused?

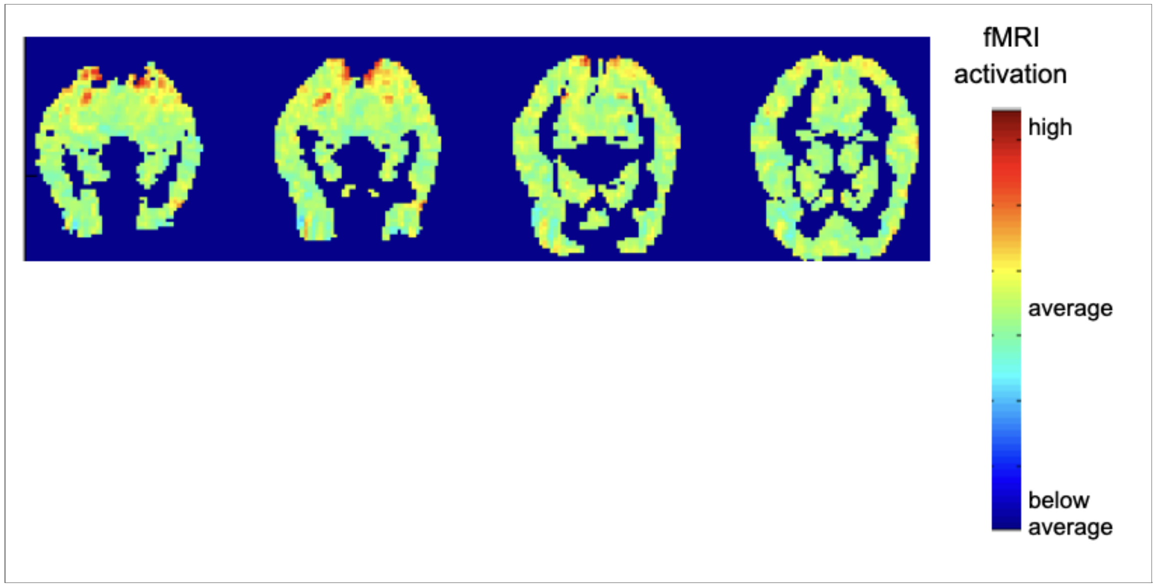


Stimulus for the study



60 distinct exemplars, presented 6 times each
Mitchell et al. Science 2008, data available **online**.

Continuous x_i



Y is the mental state (reading "house" or "bottle")
X_i are the voxel activities (voxel = volume pixel).

Continuous X_i

Naïve Bayes requires $P(X_i|Y=y_k)$ but X_i is continuous:

$$P(Y=y_k|X_1,\dots,X_d) = \frac{P(Y=y_k)\prod_i P(X_i|Y=y_k)}{\sum_\ell (Y=y_\ell)\prod_i P(X_i|Y=y_\ell)}$$

What can we do?

Continuous X_i

Naïve Bayes requires $P(X_i|Y=y_k)$ but X_i is continuous:

$$P(Y=y_k|X_1,\ldots,X_d) = \frac{P(Y=y_k)\prod_i P(X_i|Y=y_k)}{\sum_\ell (Y=y_\ell)\prod_i P(X_i|Y=y_\ell)}$$

What can we do?
Common approach: assume $P(X_i|Y=y_k)$ follows a Normal (Gaussian) distribution

$$p(X_i=x|Y=y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\exp\left(-\frac{1}{2}\frac{(x_i-\mu_{ik})^2}{\sigma_{ik}^2}\right)$$

Sometimes assume standard deviation

- is independent of Y (i.e., σ_i),
- or independent of Xi (i.e., σ_k)
- or both (i.e., σ)

Gaussian Naïve Bayes Algorithm – continuous X_i (but still discrete Y)

- Training:
    - Estimate $\pi_k \equiv P(Y = y_k)$
    - Each label y_k:
        - For each variable X_i estimate $P(X_i = x_{ij} | Y = y_k)$:
            - estimate class conditional mean μ_{ik} and standard deviation σ_{ik}
- Prediction: Classify Y^{new}

$$\begin{aligned} Y^{\text{new}} &= \underset{y_k}{\operatorname{argmax}} \, P(Y = y_k) \prod_i P(X_i^{\text{new}} = x_j^{\text{new}} | Y = y_k) && (18) \\ &= \underset{y_k}{\operatorname{argmax}} \, \pi_k \prod_i \mathcal{N}(X_i^{\text{new}}; \mu_{ik}, \sigma_{ik}) && (19) \end{aligned}$$

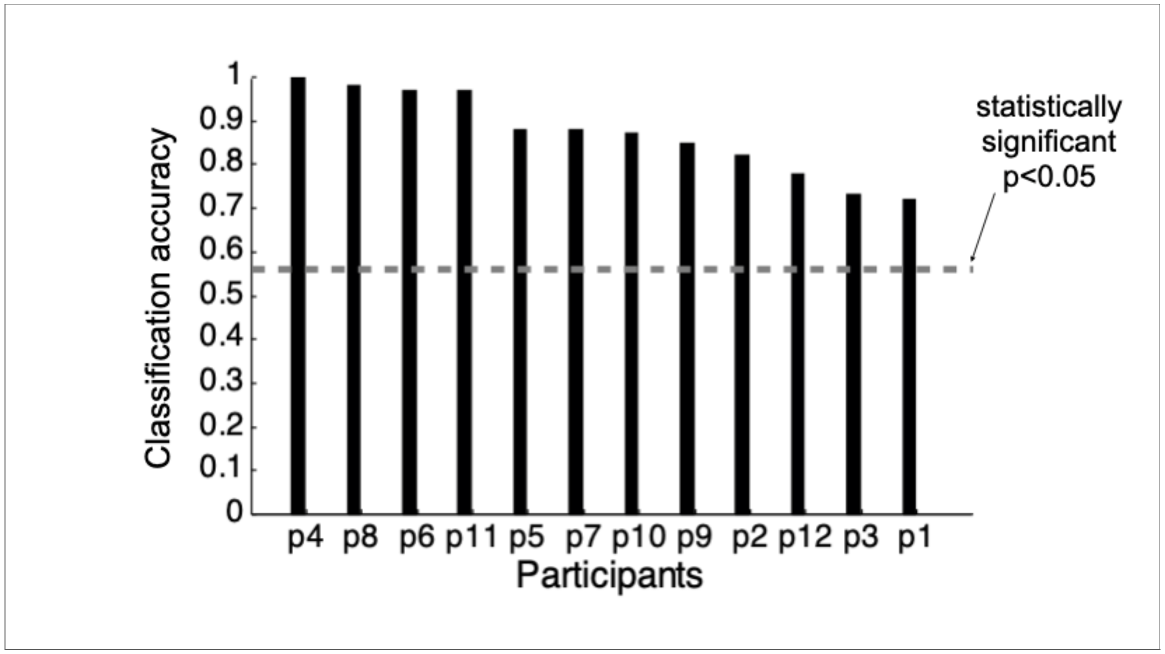
Estimating Parameters: Y discrete, X_i continuous

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k) \tag{20}$$

- i: index of feature
- j: index of data point
- k: index of class
- δ function is 1 if argument is true and 0 otherwise

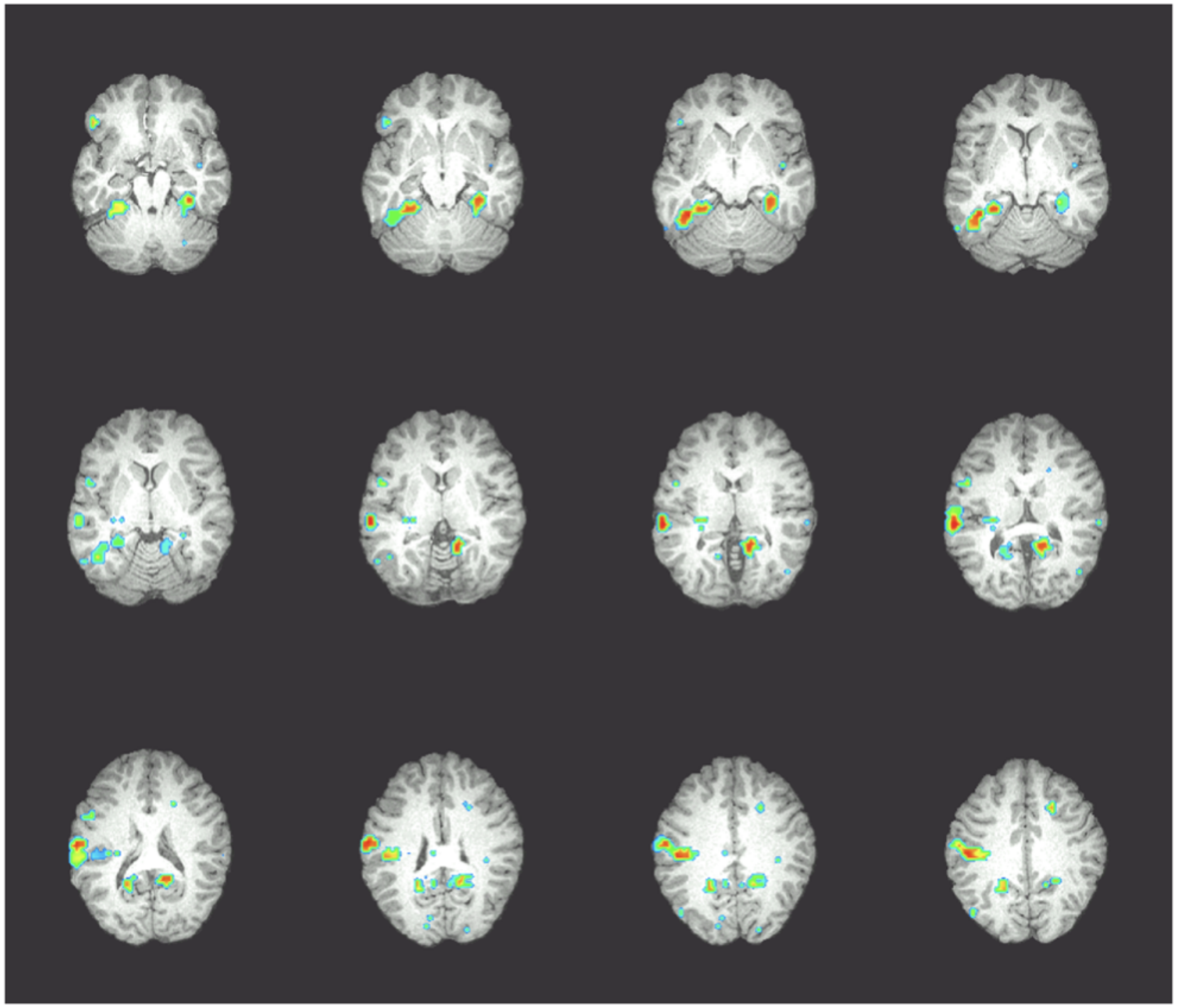
$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k) \tag{21}$$

Classification task: is person viewing a “tool” or “building”?



Where is information encoded in the brain?

Accuracies of cubical 27-voxel classifiers centered at each significant voxel [0.7-0.8]



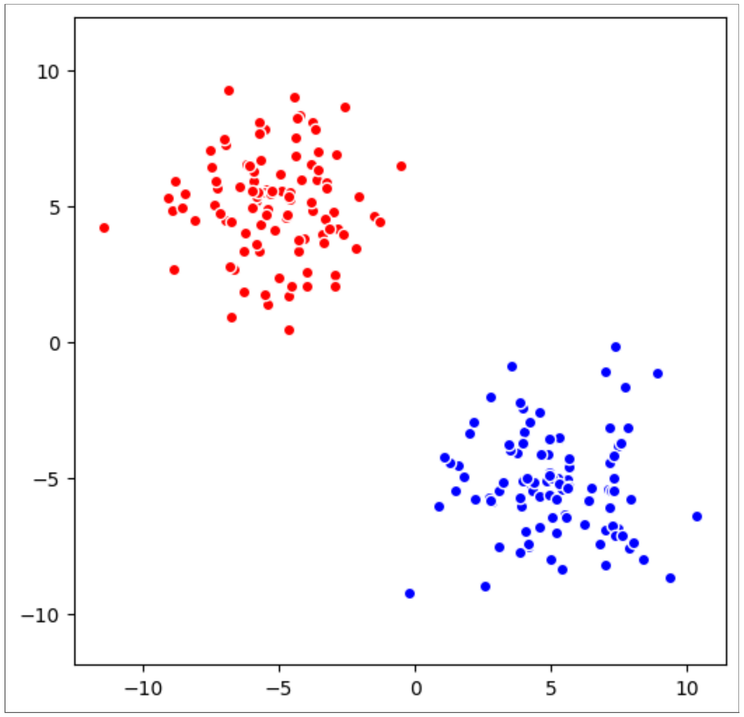
Let's simulate the behavior of GNB!

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.stats import norm

x1 = np.linspace(-10,10,1000)
x2 = np.linspace(-10,10,1000)

# Assume I know the true parameters, this is not the case usually!
mu_1_1 = -5; sigma_1_1 = 2
mu_2_1 = 5; sigma_2_1 = 2
mu_1_0 = 5; sigma_1_0 = 2
mu_2_0 = -5; sigma_2_0 = 2

# Sample data from these distributions
X_positive = norm.rvs(loc=[mu_1_1,mu_2_1], scale=[sigma_1_1, sigma_2_1], size = (100,2))
X_negative = norm.rvs(loc=[mu_1_0,mu_2_0], scale=[sigma_1_0, sigma_2_0], size = (100,2))
```

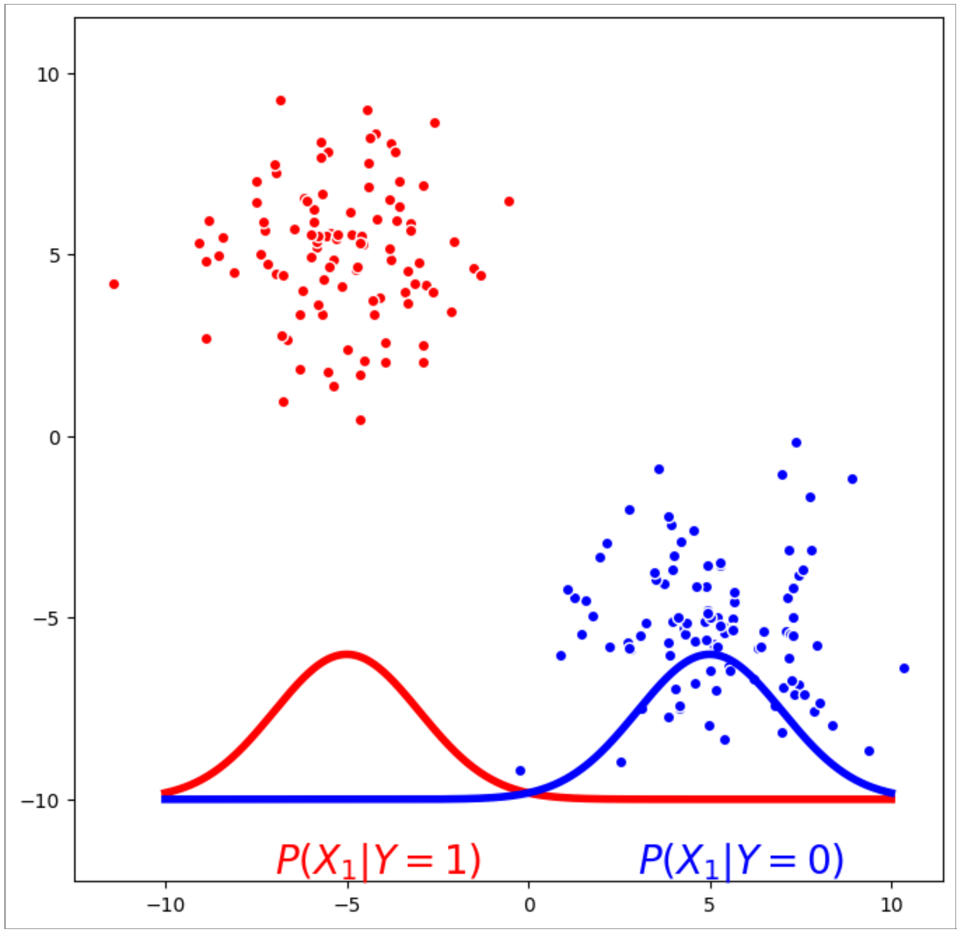
In [3]:

```
plt.figure(figsize=(6,6))
plt.scatter(X_positive[:, 0], X_positive[:, 1],facecolors='r', edgecolors='w')
plt.scatter(X_negative[:, 0], X_negative[:, 1],facecolors='b', edgecolors='w')
plt.axis([-10,10,-10,10]), plt.axis('equal');
```


10
5
0
−5
−10
−10
−5
0
5
10

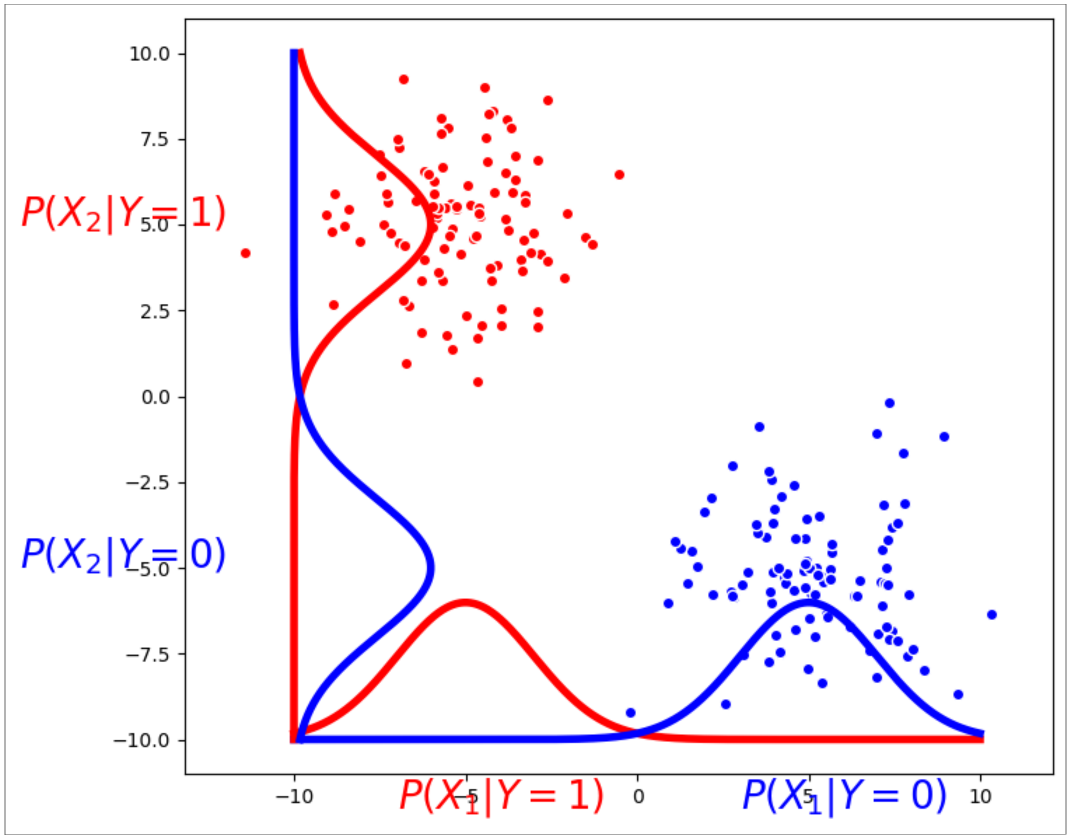
In [4]:

```
P_X1_1 = norm.pdf(x1,mu_1_1,sigma_1_1)
P_X2_1 = norm.pdf(x1,mu_2_1,sigma_2_1)
P_X1_0 = norm.pdf(x1,mu_1_0,sigma_1_0)
P_X2_0 = norm.pdf(x1,mu_2_0,sigma_2_0)

plt.figure(figsize=(8,8))
plt.scatter(X_positive[:, 0], X_positive[:, 1],facecolors='r', edgecolors='w')
plt.scatter(X_negative[:, 0], X_negative[:, 1],facecolors='b', edgecolors='w')
lim_plot = 10
plt.plot(x1,P_X1_1*2*lim_plot-lim_plot,'r',linewidth=4)
plt.text(-7, -12, r'$P(X_1|Y=1)$', color = 'red',fontsize=20)
plt.plot(x1,P_X1_0*2*lim_plot-lim_plot,'b',linewidth=4)
plt.text(3, -12, r'$P(X_1|Y=0)$', color = 'blue',fontsize=20)
plt.axis([-10,10,-10,10]), plt.axis('equal');
```


$P(X_1|Y=1)$
$P(X_1|Y=0)$
10
5
0
−5
−10
−10
−5
0
5
10

In [5]:

```
plt.figure(figsize=(8,7))
plt.scatter(X_positive[:, 0], X_positive[:, 1],facecolors='r', edgecolors='w')
plt.scatter(X_negative[:, 0], X_negative[:, 1],facecolors='b', edgecolors='w')

lim_plot = 10
plt.plot(x1,P_X1_1*2*lim_plot-lim_plot,'r',linewidth=4)
plt.text(-7, -12, r'$P(X_1|Y=1)$', color = 'red',fontsize=20)
plt.plot(x1,P_X1_0*2*lim_plot-lim_plot,'b',linewidth=4)
plt.text(3, -12, r'$P(X_1|Y=0)$', color = 'blue',fontsize=20)
plt.plot(P_X2_1*2*lim_plot-lim_plot,x1,'r',linewidth=4)
plt.text(-18,5,  r'$P(X_2|Y=1)$', color = 'red',fontsize=20)
plt.plot(P_X2_0*2*lim_plot-lim_plot,x1,'b',linewidth=4)
plt.text(-18,-5,  r'$P(X_2|Y=0)$', color = 'blue',fontsize=20)
plt.axis([-lim_plot,lim_plot,-lim_plot,lim_plot]), plt.axis('equal');
```


$P(X_2|Y=1)$
$P(X_2|Y=0)$
$P(X_1|Y=1)$
$P(X_1|Y=0)$
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10
−5
0
5
10

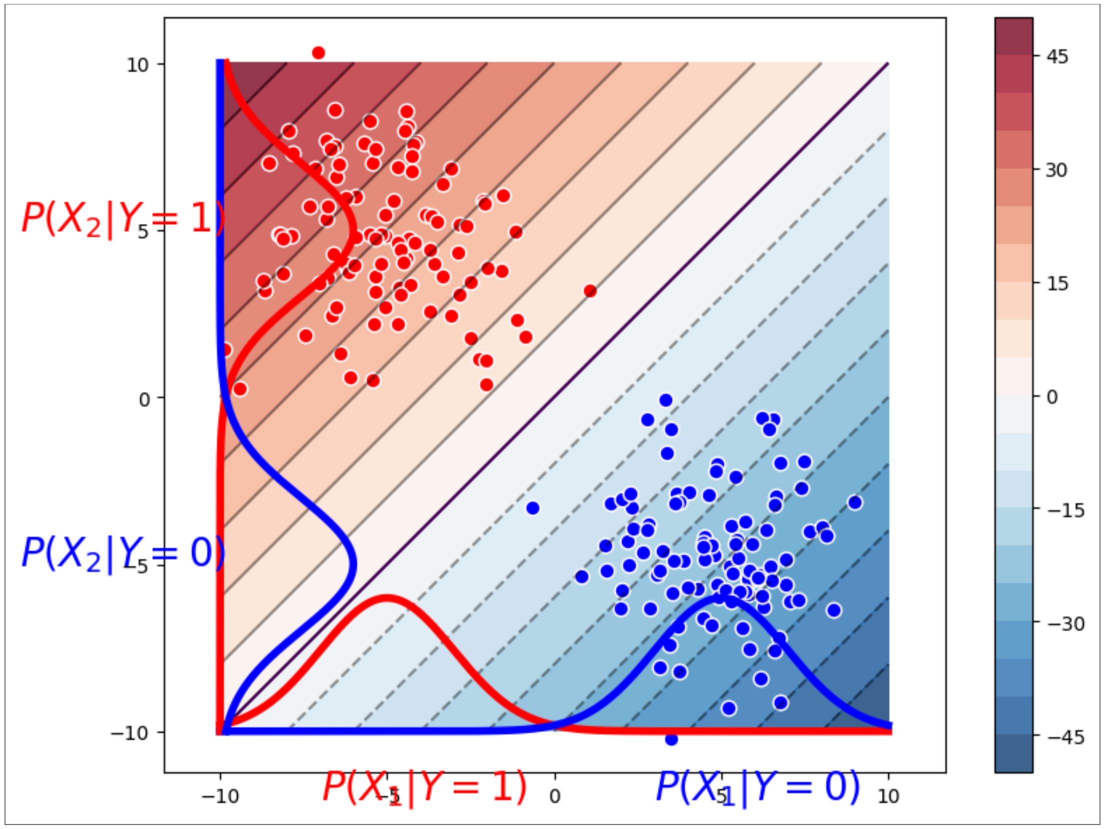
In [5]:

```
# Compute log( P(Y=1|X)/ P(Y =0|X))
# as log( P(Y=1)P(X1|Y=1)P(X2|Y=1)  / P(Y =0|X))P(X1|Y=0)P(X2|Y=0) )
# Using the real parameters. Usually, we have to estimate these!
X1,X2 = np.meshgrid(x1, x2)
def ratio_log(X1,X2):
    pY0 =0.5; pY1 = 1- pY0
    pY1pXY1 = pY1*norm.pdf(X1,mu_1_1,sigma_1_1)*norm.pdf(X2,mu_2_1,sigma_2_1)
    pY0pXY0 = pY0*norm.pdf(X1,mu_1_0,sigma_1_0)*norm.pdf(X2,mu_2_0,sigma_2_0)
    return np.log(pY1pXY1/pY0pXY0)
fX = ratio_log(X1,X2)
```

In [6]:

```
plt.figure(figsize=(9,7))
# plot contour plot
cs = plt.contourf(X1, X2, fX,20,cmap='RdBu_r',alpha=0.8);
plt.colorbar()
contours = plt.contour(cs, colors='k',alpha=0.4) # this redraws the lines in black
plt.contour(contours,levels=[0],linewidth=5)   # this makes the 0 line wider
# previous stuff
plt.scatter(X_positive[:, 0], X_positive[:, 1],facecolors='r', edgecolors='w',s=60)
plt.scatter(X_negative[:, 0], X_negative[:, 1],facecolors='b', edgecolors='w',s=60)

lim_plot = 10
plt.plot(x1,P_X1_1*2*lim_plot-lim_plot,'r',linewidth=4)
plt.text(-7, -12, r'$P(X_1|Y=1)$', color = 'red',fontsize=20)
plt.plot(x1,P_X1_0*2*lim_plot-lim_plot,'b',linewidth=4)
plt.text(3, -12, r'$P(X_1|Y=0)$', color = 'blue',fontsize=20)
plt.plot(P_X2_1*2*lim_plot-lim_plot,x1,'r',linewidth=4)
plt.text(-16,5,  r'$P(X_2|Y=1)$', color = 'red',fontsize=20)
plt.plot(P_X2_0*2*lim_plot-lim_plot,x1,'b',linewidth=4)
plt.text(-16,-5,  r'$P(X_2|Y=0)$', color = 'blue',fontsize=20)
plt.axis([-lim_plot,lim_plot,-lim_plot,lim_plot]), plt.axis('equal');
```

```
/var/folders/hx/t6_xnh5978s_6dsf2vb0kwv80000gn/T/ipykernel_13598/734830527.py:6: UserWarning: The fol
lowing kwargs were not used by contour: 'linewidth'
  plt.contour(contours,levels=[0],linewidth=5)   # this makes the 0 line wider
```


$P(X_2|Y=1)$
$P(X_2|Y=0)$
$P(X_1|Y=1)$
$P(X_1|Y=0)$
10
5
0
−5
−10
−10
−5
0
5
10
45
30
15
0
−15
−30
−45

In [25]:

```
def ratio_log_updated (X1,X2,params): # just as an example, not used here
    pY0 =len(X2)/(len(X2)+len(X1)); pY1 = 1- pY0
    pY1pXY1 = pY1*norm.pdf(X1,params['mu_1_1'],params['sigma_1_1'])*norm.pdf(X2,params['mu_2_1'],params['sigma_2_1'])
    pY0pXY0 = pY0*norm.pdf(X1,params['mu_1_0'],params['sigma_1_0'])*norm.pdf(X2,params['mu_2_0'],params['sigma_2_0'])
    return np.log(pY1pXY1/pY0pXY0)
```

In [65]:

```
def plot_GNB(X_positive,X_negative,params):
    pY0 =0.5; pY1 = 1- pY0
    P_X1_1 = norm.pdf(x1,params['mu_1_1'],params['sigma_1_1'])
    P_X2_1 = norm.pdf(x1,params['mu_2_1'],params['sigma_2_1'])
    P_X1_0 = norm.pdf(x1,params['mu_1_0'],params['sigma_1_0'])
    P_X2_0 = norm.pdf(x1,params['mu_2_0'],params['sigma_2_0'])
    X1,X2 = np.meshgrid(x1, x2)
    fX = ratio_log_updated(X1,X2,params)
    plt.figure(figsize=(10,8))
    # plot contour plot
    vmax = np.max(np.abs(fX))
    cs = plt.contourf(X1, X2, fX,20,cmap='RdBu_r',alpha=0.8, vmin = -vmax, vmax=vmax);
    plt.colorbar()
    contours = plt.contour(cs, colors='k',alpha=0.4)
    plt.contour(contours,levels=[0],linewidth=5)
    plt.scatter(X_positive[:, 0],X_positive[:, 1],facecolors='r',edgecolors='w',s=60)
    plt.scatter(X_negative[:, 0],X_negative[:, 1],facecolors='b',edgecolors='w',s=60)
    lim_plot = 10# np.max(np.stack([X_positive,X_negative]))
    plt.plot(x1,P_X1_1*2*lim_plot-lim_plot,'r',linewidth=4)
    plt.text(-7, -12, r'$P(X_1|Y=1)$', color = 'red',fontsize=20)
    plt.plot(x1,P_X1_0*2*lim_plot-lim_plot,'b',linewidth=4)
    plt.text(3, -12, r'$P(X_1|Y=0)$', color = 'blue',fontsize=20)
    plt.plot(P_X2_1*2*lim_plot-lim_plot,x1,'r',linewidth=4)
    plt.text(-16,5,  r'$P(X_2|Y=1)$', color = 'red',fontsize=20)
    plt.plot(P_X2_0*2*lim_plot-lim_plot,x1,'b',linewidth=4)
    plt.text(-16,-5,  r'$P(X_2|Y=0)$', color = 'blue',fontsize=20)
    plt.axis([-lim_plot,lim_plot,-lim_plot,lim_plot]),plt.axis('equal')
```

The features x_1 and x_2 in the simulation where conditionally independent

What if:

- we make them dependent (use a non-diagonal covariance matrix to sample multivariate gaussian)
- We still use conditional independence as an assumption for GNB

1st: case where same variance

In [7]:

```python
from scipy.stats import multivariate_normal

# Same param as before
mu_1_1 = -5; sigma_1_1 = 2; mu_2_1 = 5; sigma_2_1 = 2
mu_1_0 = 5; sigma_1_0 = 2; mu_2_0 = -5; sigma_2_0 = 2

cov_positive = np.array([[sigma_1_1**2,3], [3,sigma_2_1**2]] )
cov_negative = np.array([[sigma_1_0**2,3], [3,sigma_2_0**2]] )

print(cov_positive)

# Sample data from these distributions
X_positive = multivariate_normal.rvs(mean=[mu_1_1,mu_2_1],cov=cov_positive,size=(100))
X_negative = multivariate_normal.rvs(mean=[mu_1_0,mu_2_0],cov=cov_negative,size=(100))
```

```
[[4 3]
 [3 4]]
```

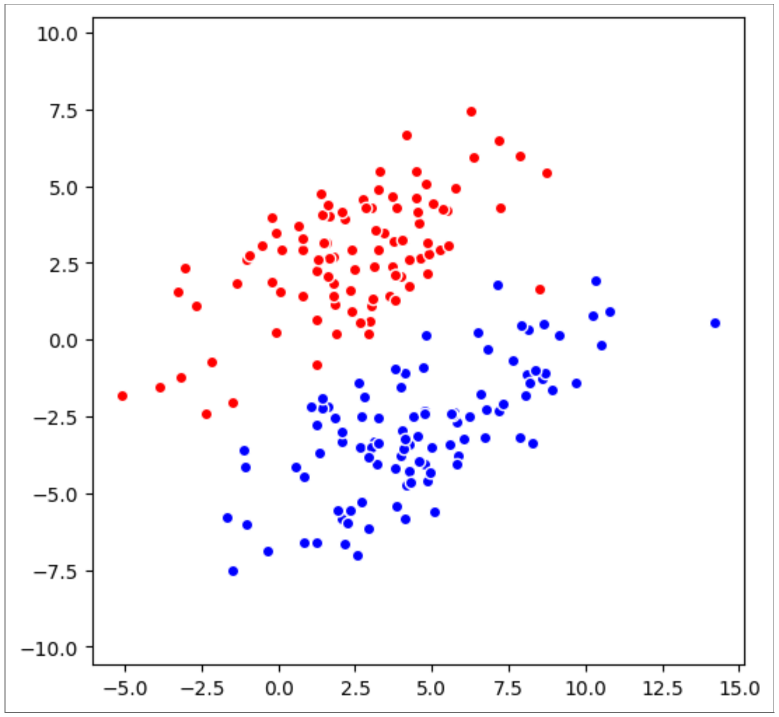
In [69]:

```
from scipy.stats import multivariate_normal

# Same param as before
mu_1_1 = 3; sigma_1_1 = 3; mu_2_1 = 3; sigma_2_1 = 2
mu_1_0 = 4.5; sigma_1_0 = 3; mu_2_0 = -3; sigma_2_0 = 2

cov_positive = np.array([[sigma_1_1**2,4], [4,sigma_2_1**2]] )
cov_negative = np.array([[sigma_1_0**2,4], [4,sigma_2_0**2]] )

print(cov_positive)

# Sample data from these distributions
X_positive = multivariate_normal.rvs(mean=[mu_1_1,mu_2_1],cov=cov_positive,size=(100))
X_negative = multivariate_normal.rvs(mean=[mu_1_0,mu_2_0],cov=cov_negative,size=(100))
```

```
[[9 4]
 [4 4]]
```

In [70]:

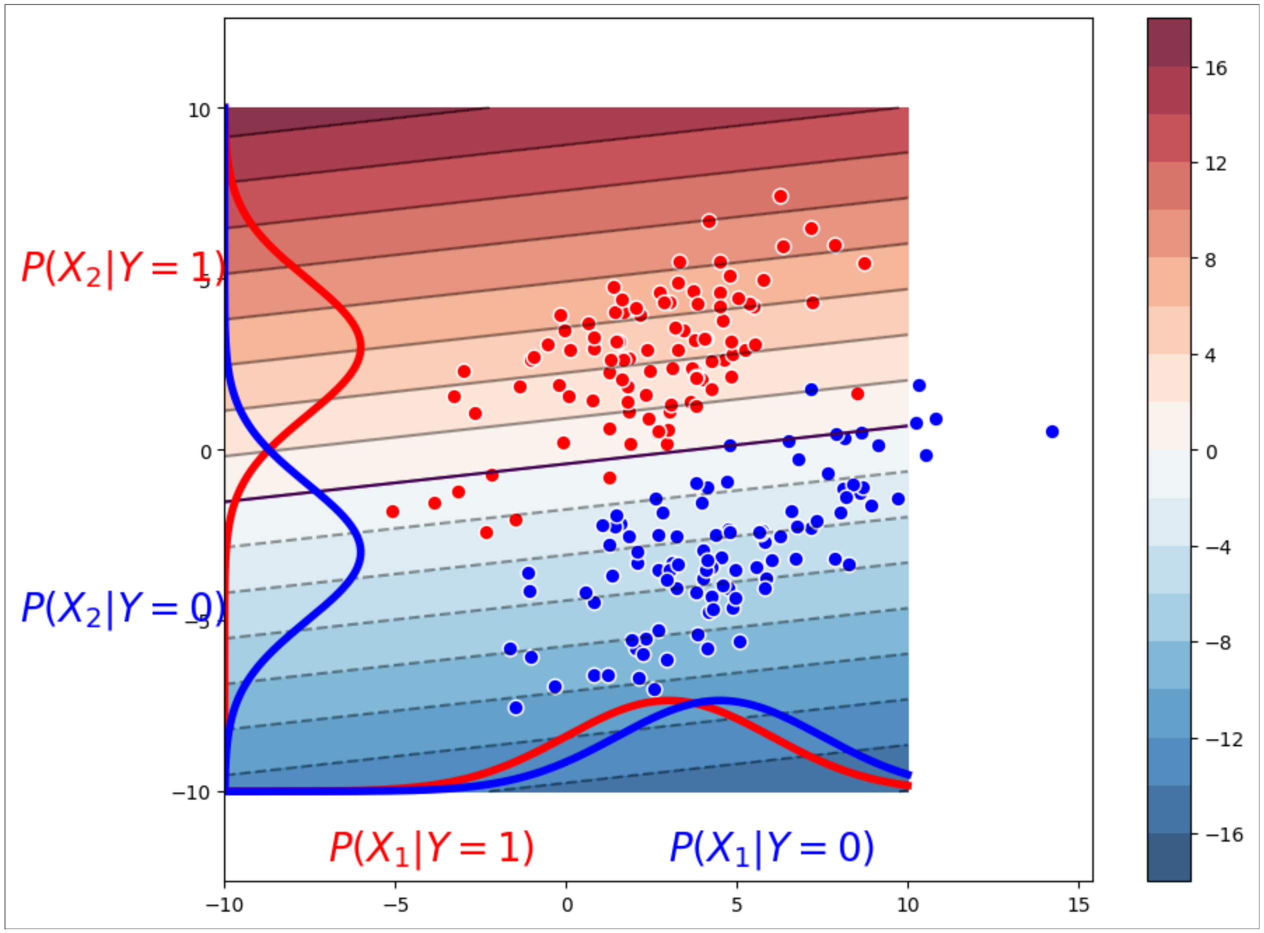
```
plt.figure(figsize=(6,6))
plt.scatter(X_positive[:, 0], X_positive[:, 1],facecolors='r', edgecolors='w')
plt.scatter(X_negative[:, 0], X_negative[:, 1],facecolors='b', edgecolors='w')
plt.axis([-10,10,-10,10]), plt.axis('equal');
```


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0

In [71]:

```
# Assume I perfectly estimate the parameters (not true for limited data!)
params = dict(mu_1_1 = mu_1_1, sigma_1_1 = sigma_1_1,mu_2_1 = mu_2_1, sigma_2_1 =sigma_2_1,
            mu_1_0 = mu_1_0, sigma_1_0 = sigma_1_0,mu_2_0 = mu_2_0, sigma_2_0 = sigma_2_0)

plot_GNB(X_positive,X_negative,params)
```

```
/var/folders/hx/t6_xnh5978s_6dsf2vb0kwv80000gn/T/ipykernel_27023/3863331813.py:15: UserWarning: The f
ollowing kwargs were not used by contour: 'linewidth'
  plt.contour(contours,levels=[0],linewidth=5)
```



In [28]:

```
# Estimate

mu_1_1, mu_2_1 = np.mean(X_positive,axis=0)
mu_1_0, mu_2_0 = np.mean(X_negative,axis=0)

# Same Variance!

sigma_1_1, sigma_2_1 = np.std(X_positive,axis=0)
sigma_1_0, sigma_2_0 = np.std(X_negative,axis=0)
print(sigma_1_1, sigma_2_1)
print(sigma_1_0, sigma_2_0)
```

```
3.358161442980908 0.6031494103134225
3.1197053968679875 0.6581223760963012
```

Is GNB a linear separator?

- It depends on whether we allow it to learn different standard deviations for each class. Decision rule:

$$\ln \frac{P(Y=1|X_1 \ldots X_d)}{P(Y=0|X_1 \ldots X_d)} = \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{P(X_i|Y=1)}{P(X_i|Y=0)} \quad > \text{ or } < 0? \tag{22}$$

- If X_is are $\mathcal{N}(\mu_{ik}, \sigma_{ik})$:

$$p(X_i = x|Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} \exp\left(-\frac{1}{2}\frac{(x_i - \mu_{ik})^2}{\sigma_{ik}^2}\right)$$

Is GNB a linear separator?

$$\ln \frac{P(Y=1|X_1 \dots X_d)}{P(Y=0|X_1 \dots X_d)} = \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{P(X_i|Y=1)}{P(X_i|Y=0)} \tag{23}$$

$$= \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{\sigma_{i0}}{\sigma_{i1}} + G(X) \tag{24}$$

$$G(X) = \sum_i \ln \exp\big(- \frac{1}{2} \frac{(x_i - \mu_{i1})^2}{\sigma_{i1}^2} + \frac{1}{2} \frac{(x_i - \mu_{i0})^2}{\sigma_{i0}^2}\big) \tag{25}$$

$$= -\frac{1}{2} \sum_i \big(x_i^2 (\frac{1}{\sigma_{i1}^2} - \frac{1}{\sigma_{i0}^2}) - 2x_i (\frac{\mu_{i1}}{\sigma_{i1}^2} - \frac{\mu_{i0}}{\sigma_{i0}^2}) + (\frac{\mu_{i1}^2}{\sigma_{i1}^2} - \frac{\mu_{i0}^2}{\sigma_{i0}^2})\big) \tag{26}$$

What happens if we force $\sigma_{i0} = \sigma_{i1}$?

- We get a linear decision boundary. Otherwise, it's a quadratic decision boundary.

In [29]:

```
# Same param as before
mu_1_1 = -5; sigma_1_1 = 2
mu_2_1 = 5; sigma_2_1 = 2
mu_1_0 = 5; sigma_1_0 = 2
mu_2_0 = -5; sigma_2_0 = 2

cov_positive = np.array([[sigma_1_1**2,3], [3,sigma_2_1**2]] )
cov_negative = np.array([[sigma_1_0**2,3], [3,sigma_2_0**2]] )

# Sample data from these distributions
X_positive = multivariate_normal.rvs(mean=[mu_1_1,mu_2_1], cov=cov_positive, size = (30))
X_negative = multivariate_normal.rvs(mean=[mu_1_0,mu_2_0], cov=cov_negative, size = (30))
```

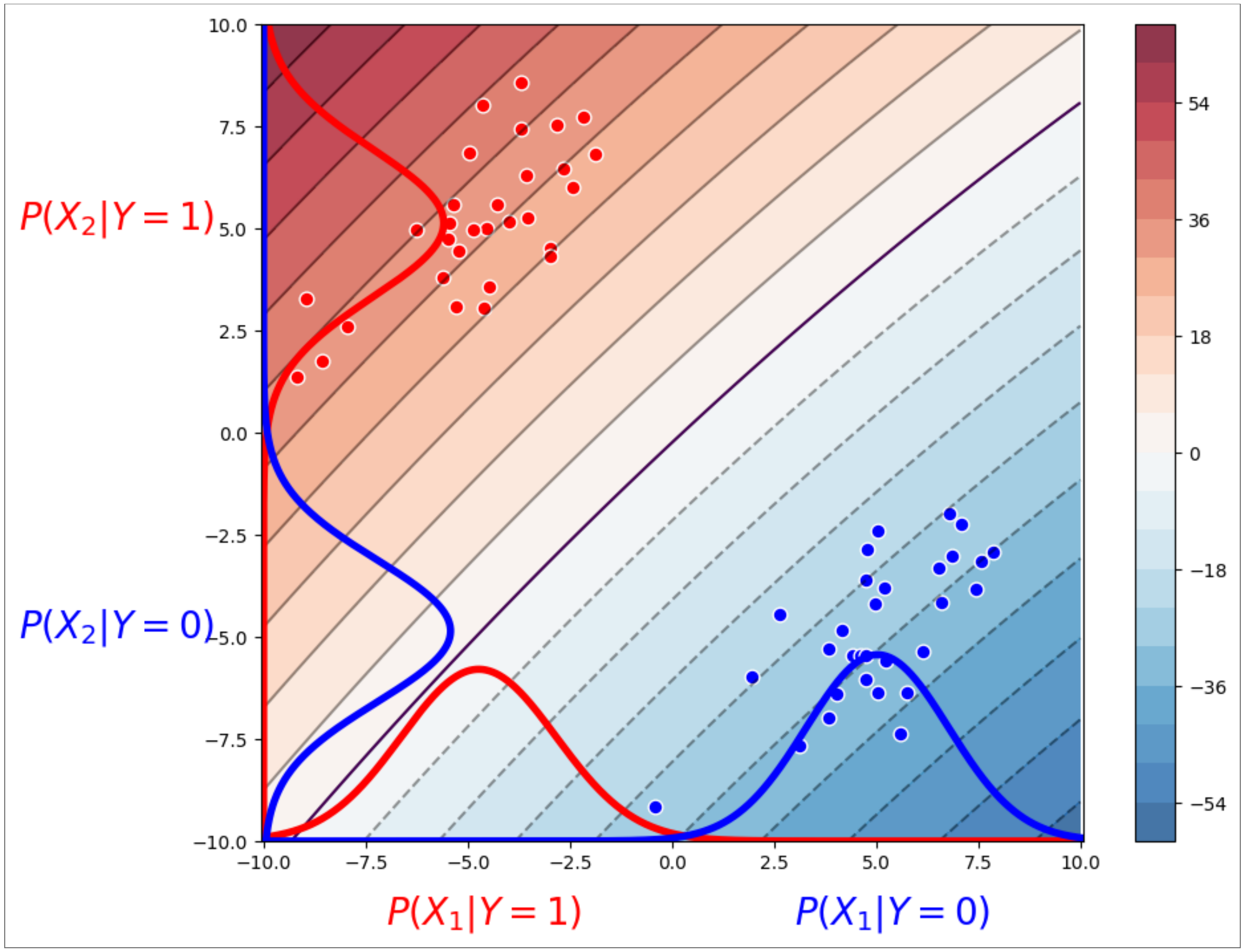
In [30]:

```
params = dict()
# Estimate - Different variance because of limited sample size
params['mu_1_1'], params['mu_2_1'] = np.mean(X_positive,axis=0)
params['sigma_1_1'], params['sigma_2_1'] = np.std(X_positive,axis=0)
params['mu_1_0'], params['mu_2_0'] = np.mean(X_negative,axis=0)
params['sigma_1_0'], params['sigma_2_0'] = np.std(X_negative,axis=0)
```

In [31]:

```
plot_GNB(X_positive,X_negative,params)
```

```
/var/folders/hx/t6_xnh5978s_6dsf2vb0kwv80000gn/T/ipykernel_27023/3863331813.py:15: UserWarning: The f
ollowing kwargs were not used by contour: 'linewidth'
  plt.contour(contours,levels=[0],linewidth=5)
```

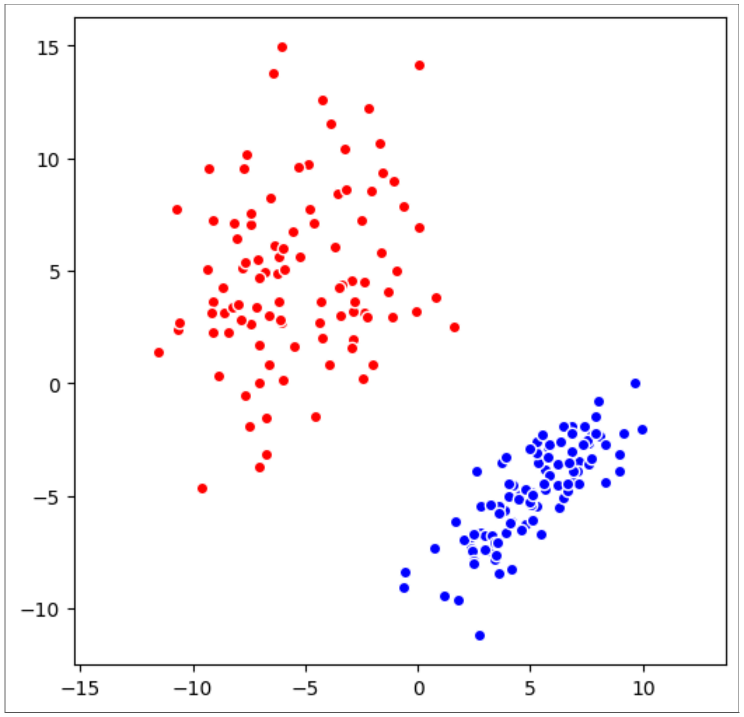


In [32]:

```python
# Let's set up another example in which the variances are actually different
mu_1_1 = -5; sigma_1_1 = 3
mu_2_1 = 5; sigma_2_1 = 4
mu_1_0 = 5; sigma_1_0 = 2
mu_2_0 = -5; sigma_2_0 = 2

cov_positive = np.array([[sigma_1_1**2,3], [3,sigma_2_1**2]] )
cov_negative = np.array([[sigma_1_0**2,3], [3,sigma_2_0**2]] )

# Sample data from these distributions
X_positive = multivariate_normal.rvs(mean=[mu_1_1,mu_2_1],cov=cov_positive,size=(100))
X_negative = multivariate_normal.rvs(mean=[mu_1_0,mu_2_0],cov=cov_negative,size=(100))

plt.figure(figsize=(6,6))

plt.scatter(X_positive[:, 0], X_positive[:, 1],facecolors='r', edgecolors='w')
plt.scatter(X_negative[:, 0], X_negative[:, 1],facecolors='b', edgecolors='w')
plt.axis([-10,10,-10,10]),plt.axis('equal')
```

Out[32]:

```
((-10.0, 10.0, -10.0, 10.0),
 (-12.590047580156053,
  11.007571500903204,
  -12.511148931397772,
  16.251494191115942))
```


15
10
5
0
−5
−10
−15
−10
−5
0
5
10

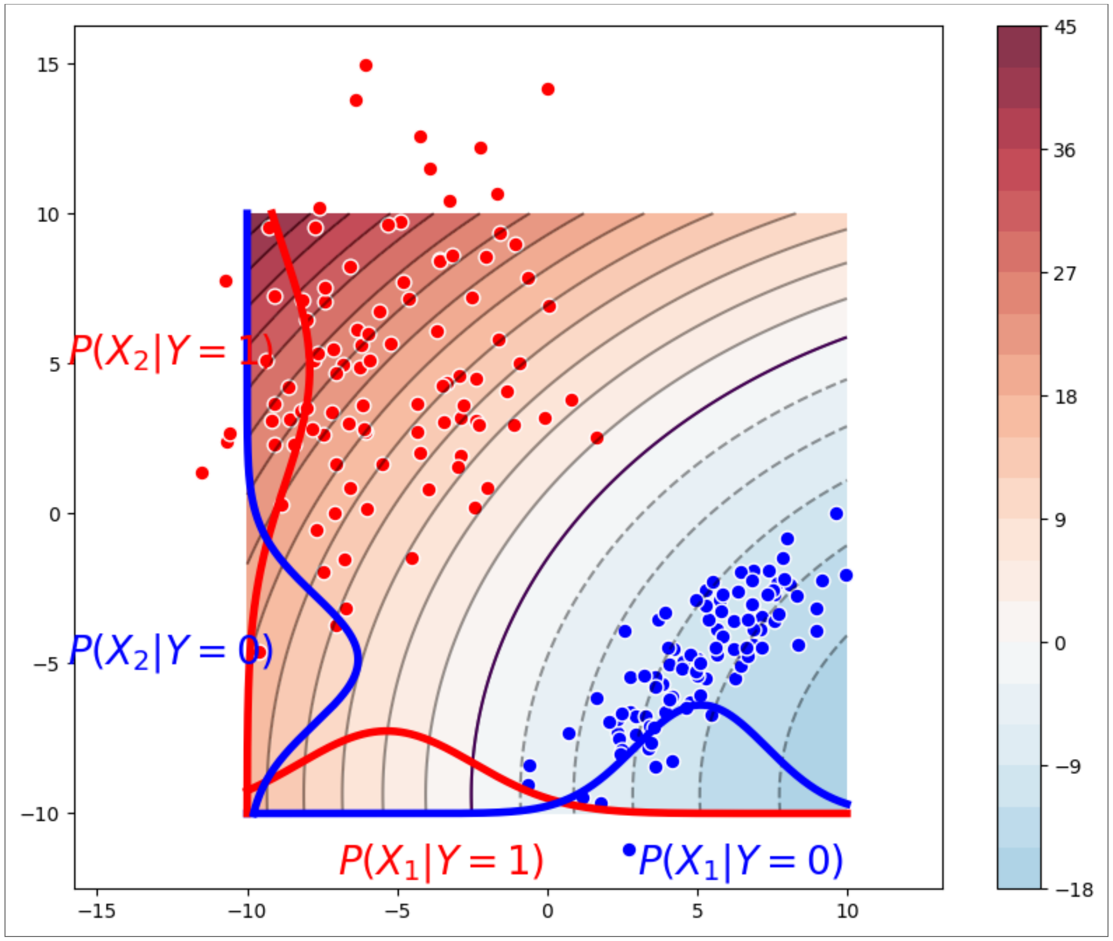
In [33]:

```
params = dict()
# Estimate - Different variance
params['mu_1_1'], params['mu_2_1'] = np.mean(X_positive,axis=0)
params['sigma_1_1'], params['sigma_2_1'] = np.std(X_positive,axis=0)
params['mu_1_0'], params['mu_2_0'] = np.mean(X_negative,axis=0)
params['sigma_1_0'], params['sigma_2_0'] = np.std(X_negative,axis=0)
```

In [34]:

```
plot_GNB(X_positive,X_negative,params)
```

```
/var/folders/hx/t6_xnh5978s_6dsf2vb0kwv80000gn/T/ipykernel_27023/3863331813.py:15: UserWarning: The f
ollowing kwargs were not used by contour: 'linewidth'
  plt.contour(contours,levels=[0],linewidth=5)
```



In [35]:

```
from sklearn import datasets

plt.figure(figsize=(5,5))
X, y = datasets.make_circles(n_samples=200, factor=.25,noise=.1)

# scale
X_positive = X[y==1]*8
X_negative = X[y==0]*8
```

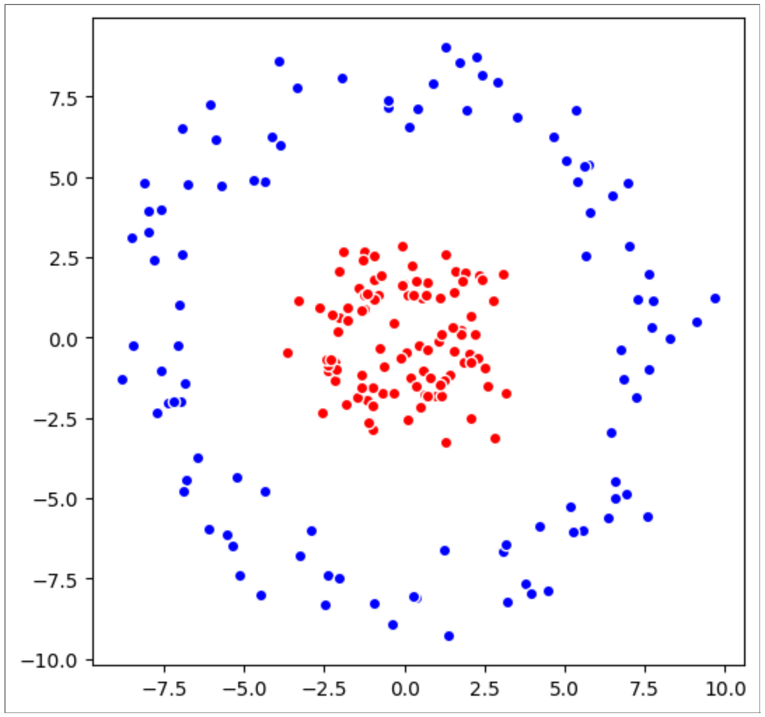
```
<Figure size 500x500 with 0 Axes>
```

In [36]:

```
plt.figure(figsize=(6,6))

plt.scatter(X_positive[:, 0], X_positive[:, 1],facecolors='r', edgecolors='w')
plt.scatter(X_negative[:, 0], X_negative[:, 1],facecolors='b', edgecolors='w')
plt.axis([-10,10,-10,10]), plt.axis('equal')
```

Out[36]:

```
((-10.0, 10.0, -10.0, 10.0),
 (-9.726813172284835,
  10.599700820055302,
  -10.203059802289701,
  9.95269893500399))
```



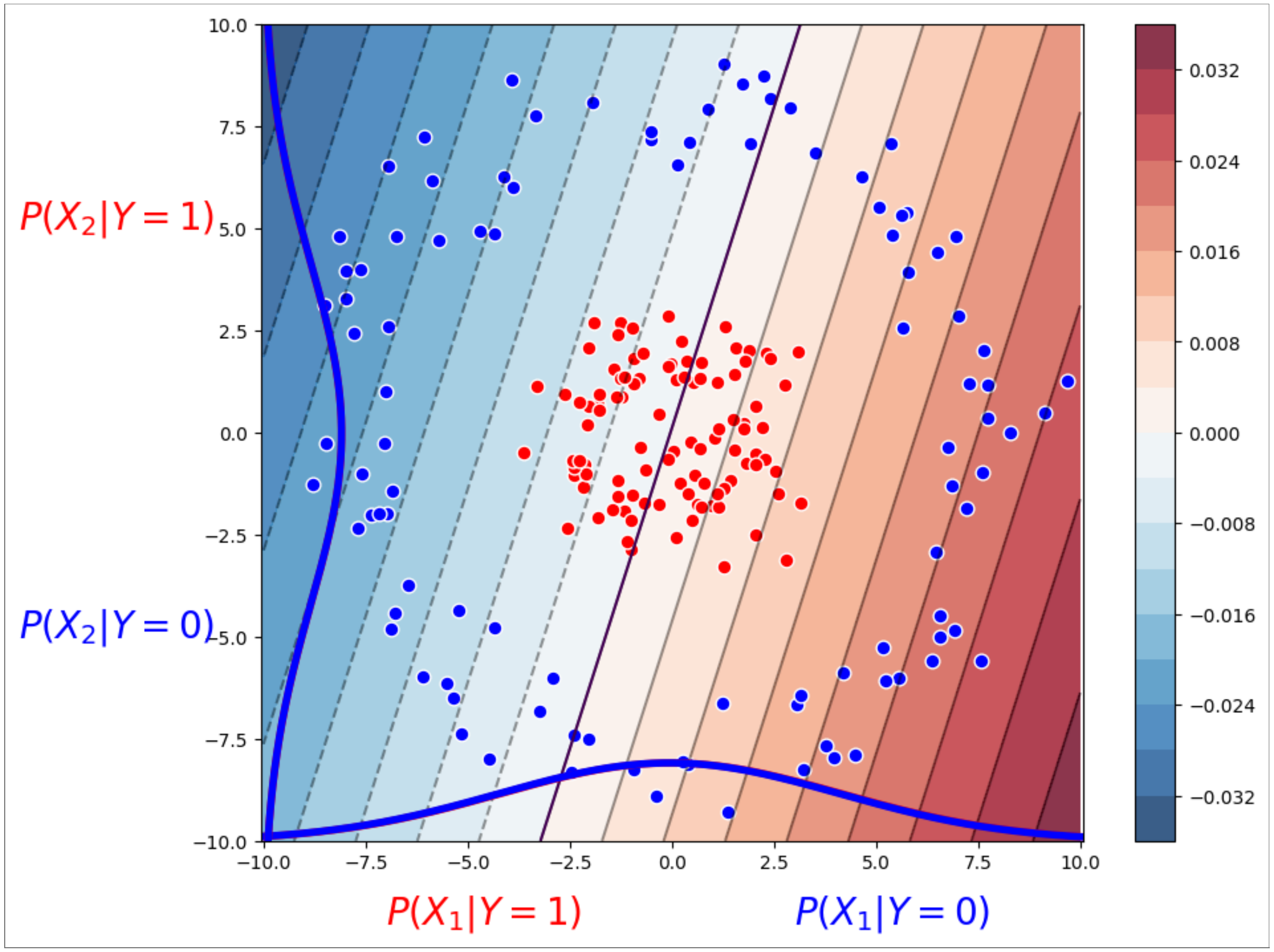
In [37]:

```
params = dict()
# Artificially force same variances
params['mu_1_1'], params['mu_2_1'] = np.mean(X_positive,axis=0)
params['sigma_1_1'], params['sigma_2_1'] = np.std(np.vstack([X_positive,X_negative]),axis=0)
params['mu_1_0'], params['mu_2_0'] = np.mean(X_negative,axis=0)
params['sigma_1_0'], params['sigma_2_0'] = np.std(np.vstack([X_positive,X_negative]),axis=0)
```

In [38]:

```
plot_GNB(X_positive,X_negative,params)
```

```
/var/folders/hx/t6_xnh5978s_6dsf2vb0kwv80000gn/T/ipykernel_27023/3863331813.py:15: UserWarning: The f
ollowing kwargs were not used by contour: 'linewidth'
  plt.contour(contours,levels=[0],linewidth=5)
```



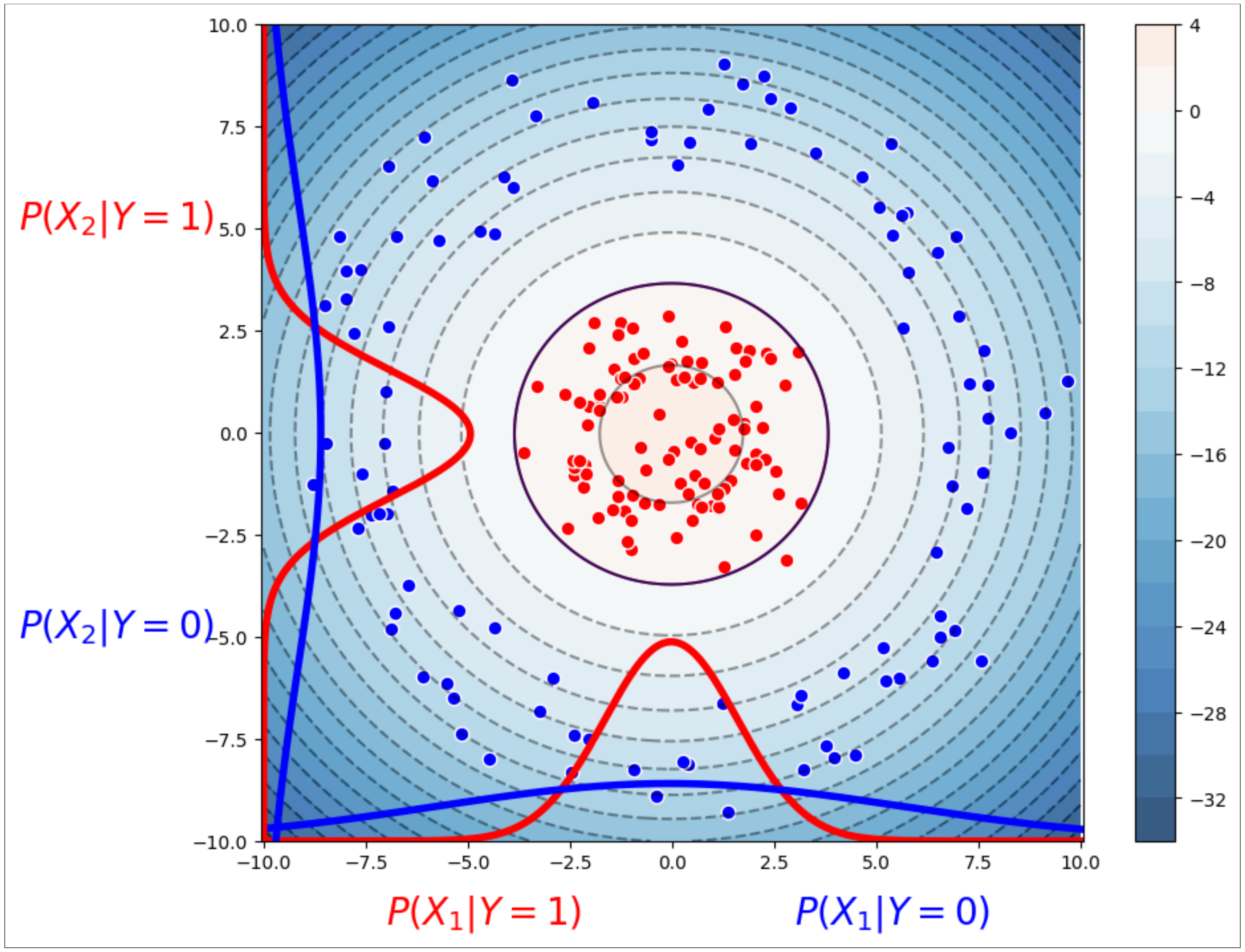
In [39]:

```
params = dict()
# Estimate - Different variance
params['mu_1_1'], params['mu_2_1'] = np.mean(X_positive,axis=0)
params['sigma_1_1'], params['sigma_2_1'] = np.std(X_positive,axis=0)
params['mu_1_0'], params['mu_2_0'] = np.mean(X_negative,axis=0)
params['sigma_1_0'], params['sigma_2_0'] = np.std(X_negative,axis=0)
```

In [40]:

```
plot_GNB(X_positive,X_negative,params)
```

```
/var/folders/hx/t6_xnh5978s_6dsf2vb0kwv80000gn/T/ipykernel_27023/3863331813.py:15: UserWarning: The f
ollowing kwargs were not used by contour: 'linewidth'
  plt.contour(contours,levels=[0],linewidth=5)
```



The last example is a case where the conditional independence assumption is incorrect

- but GNB does very well

What you should know

Naïve Bayes classifier

- What’s the assumption
- Why we use it
- How do we learn it
- The different observations we made about it
- Why is Bayesian estimation important