10-315 Introduction to Machine Learning (SCS Majors)
Lecture 6: Naive Bayes

Leila Wehbe
Carnegie Mellon University
Machine Learning Department

Reading: (http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf">) Generative and Discriminative Classifiers by Tom Mitchell.

Lecture outcomes:

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

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 estimate for each label y_k:
        - For each variable X_i:
            - For each value x_{ij} that X_i can take.

- Prediction: Classify Y^{new}

$$\begin{aligned} Y^{\text{new}} &= \underset{y_k}{\operatorname{argmax}}\ P(Y = y_k) \prod_i P(X_i^{\text{new}} = x_j^{\text{new}} | Y = y_k) \\ &= \underset{y_k}{\operatorname{argmax}}\ \pi_k \prod_i \theta_{i, X_i^{\text{new}}, k} \end{aligned}$$

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.

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

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_j 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.

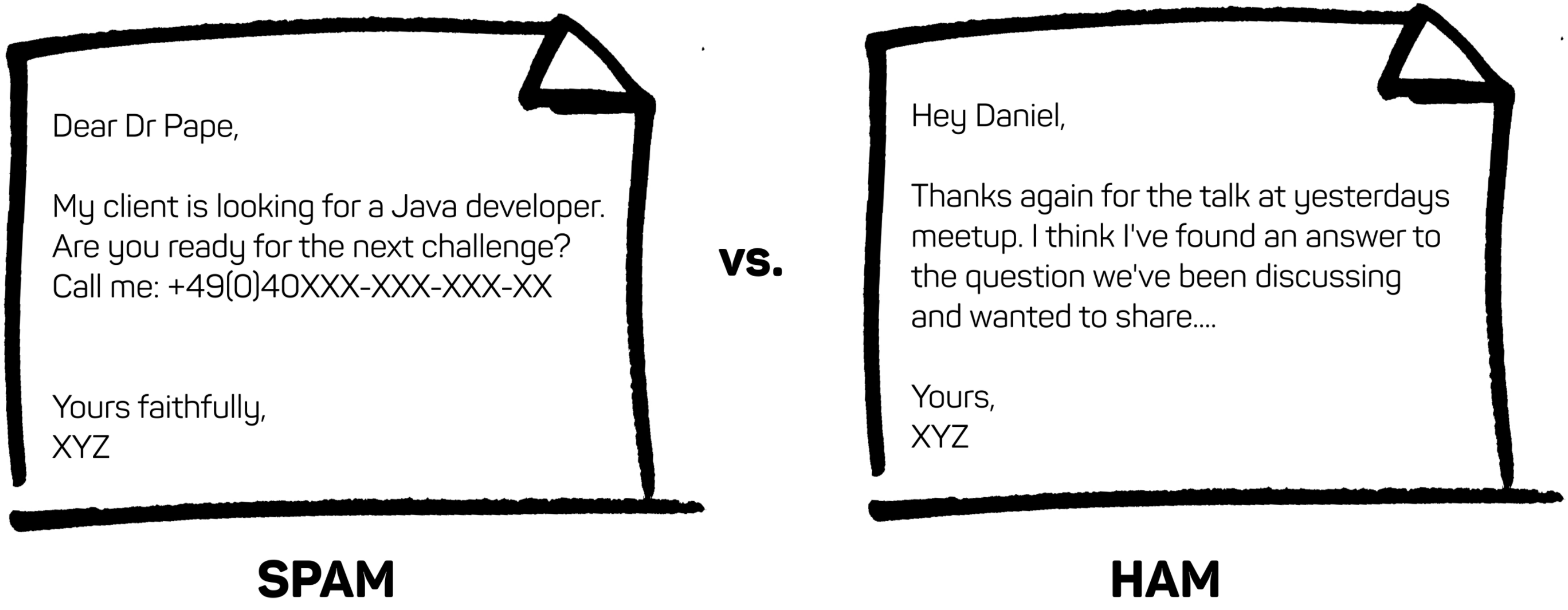
To estimate:

- 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?



[image by Daniel Pape (https://blog.codecentric.de/en/2016/06/spam-classification-using-sparks-dataframes-ml-zeppelin-part-1/)](https://blog.codecentric.de/en/2016/06/spam-classification-using-sparks-dataframes-ml-zeppelin-part-1/)

- 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 = (X1, X2, … Xn) with n the number of words in English.
- (This is what we do in homework 2)

What are the limitations with this representation?

How can we express X?

- Y discrete valued. e.g., Spam or not
- X = (X1, X2, … Xn) with n the number of words in English.
- (This is what we do in homework 2)

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
- $X = (X_1, X_2, \ldots X_d)$ = 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)

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)

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 $\boldsymbol{X_i}$s have the same distribution, we only have to estimate one parameter per word, per class.

$\boldsymbol{P(X|Y = y_k)}$ 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

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

Probability of observing a document with α_1 count of $w_1, \alpha 21$ 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:
$$P(\theta) = \frac{\theta^{\beta_\theta}\theta^{\beta_\theta}, \ldots \theta^{\beta_\theta}}{\mathrm{Beta}(\beta_\theta, \beta_\theta, \ldots, \beta_\theta)}$$

- if constant across classes but different for different words:
$$P(\theta) = \frac{\theta^{\beta_1}\theta^{\beta_2}, \ldots \theta^{\beta_J}}{\mathrm{Beta}(\beta_1, \beta_2, \ldots, \beta_J)}$$

- if different for different classes k and words:
$$P(\theta_k) = \frac{\theta^{\beta_{1k}}\theta^{\beta_{2k}}, \ldots \theta^{\beta_{Jk}}}{\mathrm{Beta}(\beta_{1k}, \beta_{2k}, \ldots, \beta_{Jk})}$$

MAP estimates for Bag of words:

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

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

Again the prior acts like halucinated examples

What βs should we choose?

Example: Twenty NewsGroups

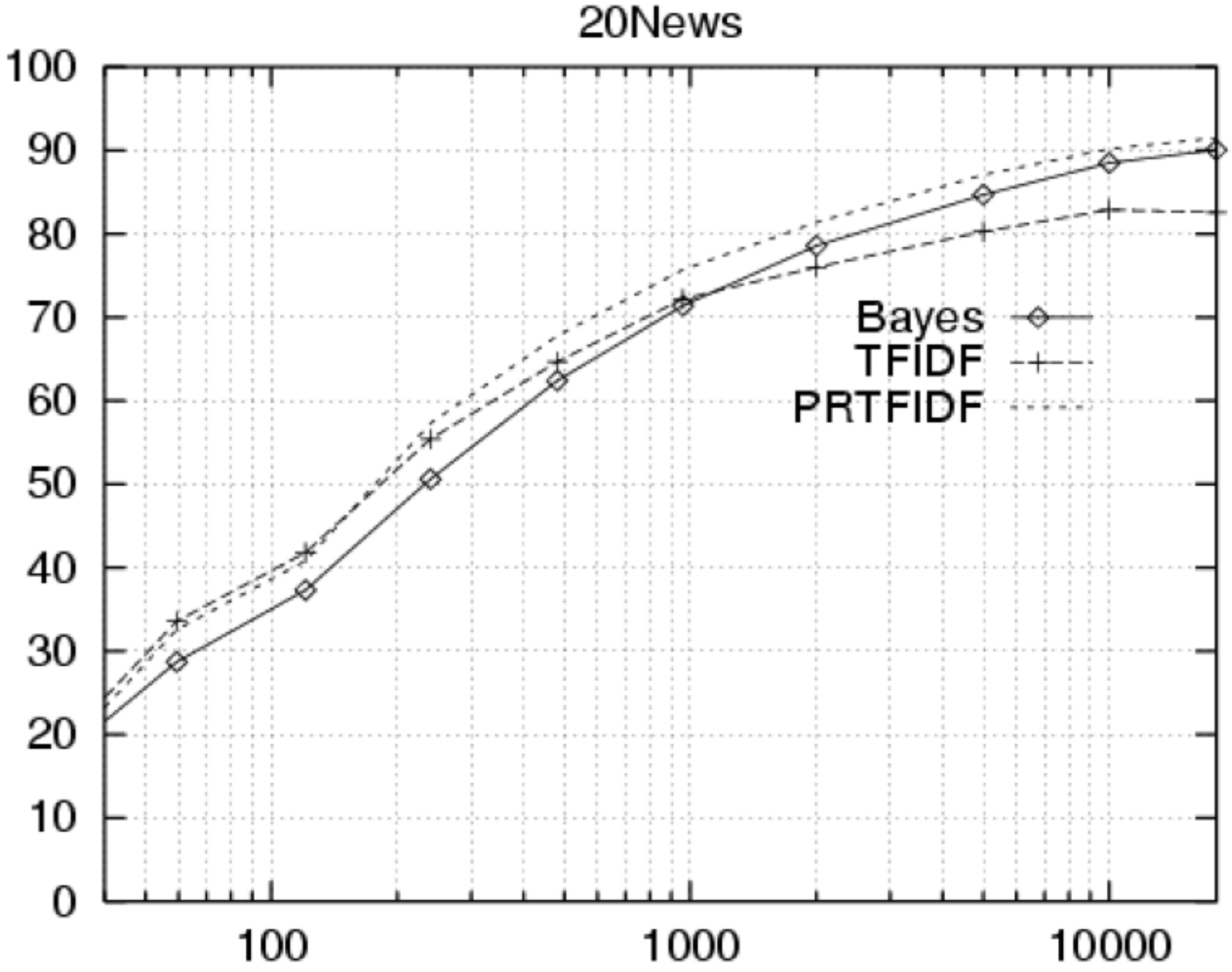
For code and data, see www.cs.cmu.edu/~tom/mlbook.html click on “Software and Data”.

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)

More recently, algorithms such as LSTMs and Transformers are able to capture the sequential aspect of language and produce more complex predictions.

- They do have many parameters, but nowhere as much as we mentioned before (10000^{1000}).

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 LSTMs and Transformers are able to capture the sequential aspect of language and produce more complex predictions.

- They do have many parameters, but nowhere as much as we mentioned before (10000^{1000}).

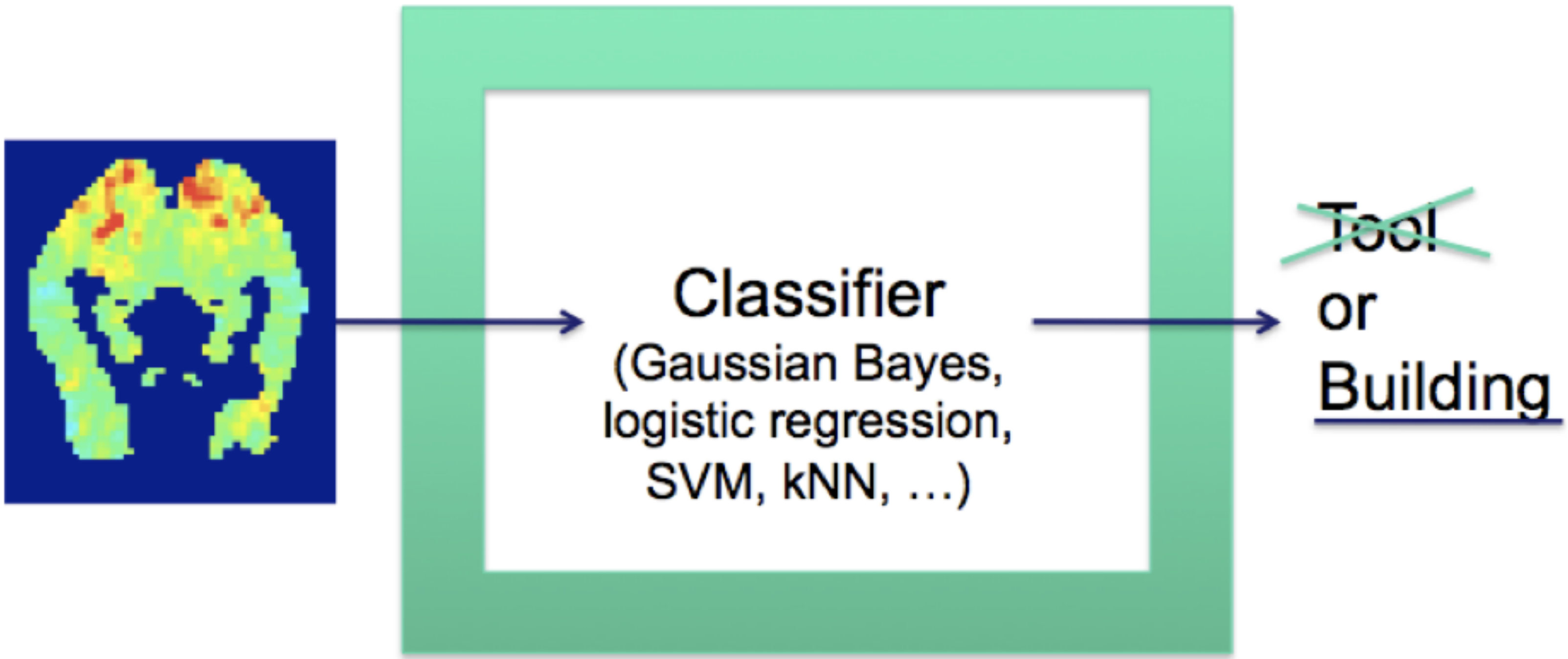
Continuous X_is

What can we do?

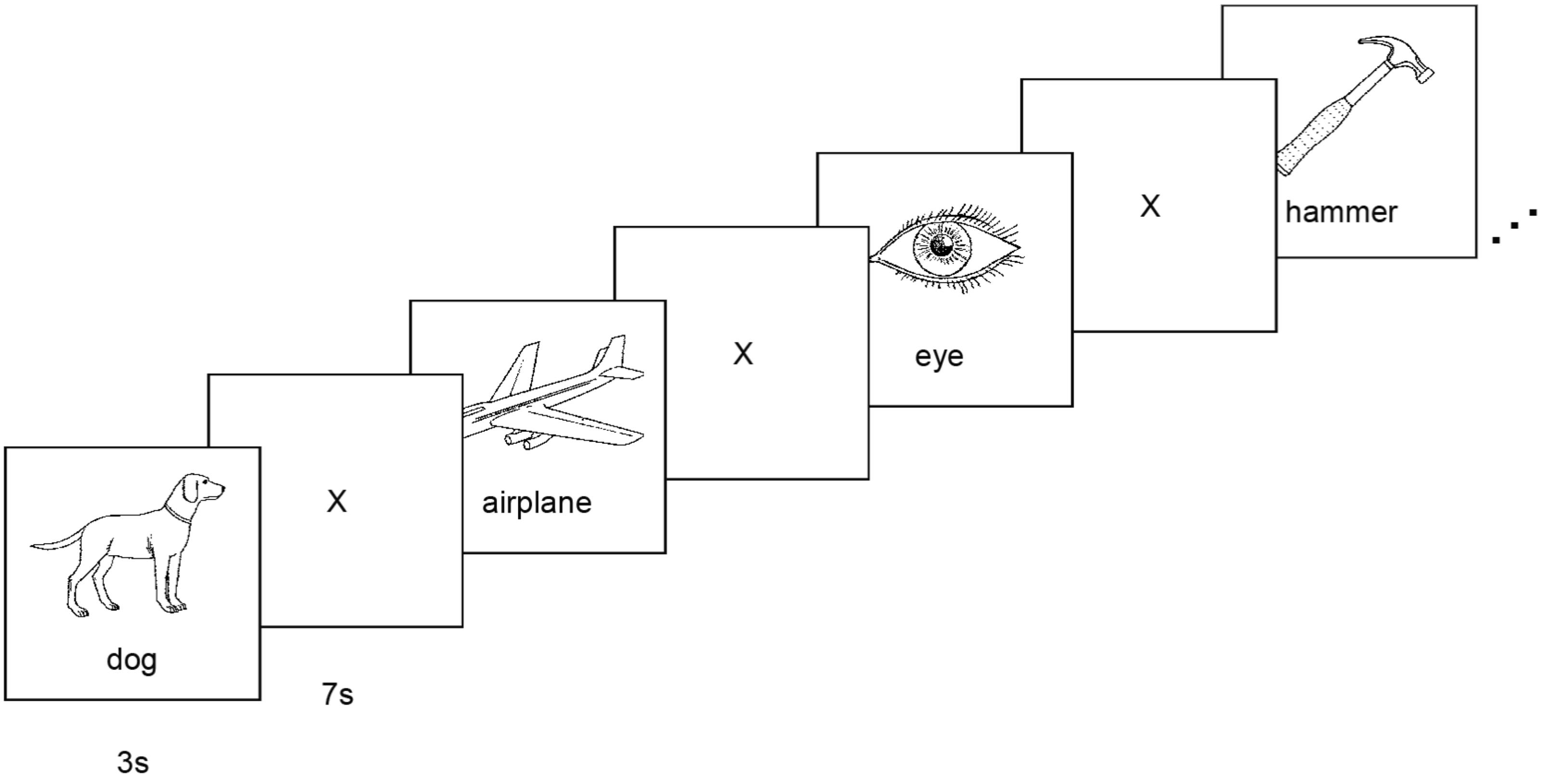
E.g. image classification, where X_i is real valued

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?


Classifier
(Gaussian Bayes, logistic regression, SVM, kNN, …)
~~Tool~~
or
Building

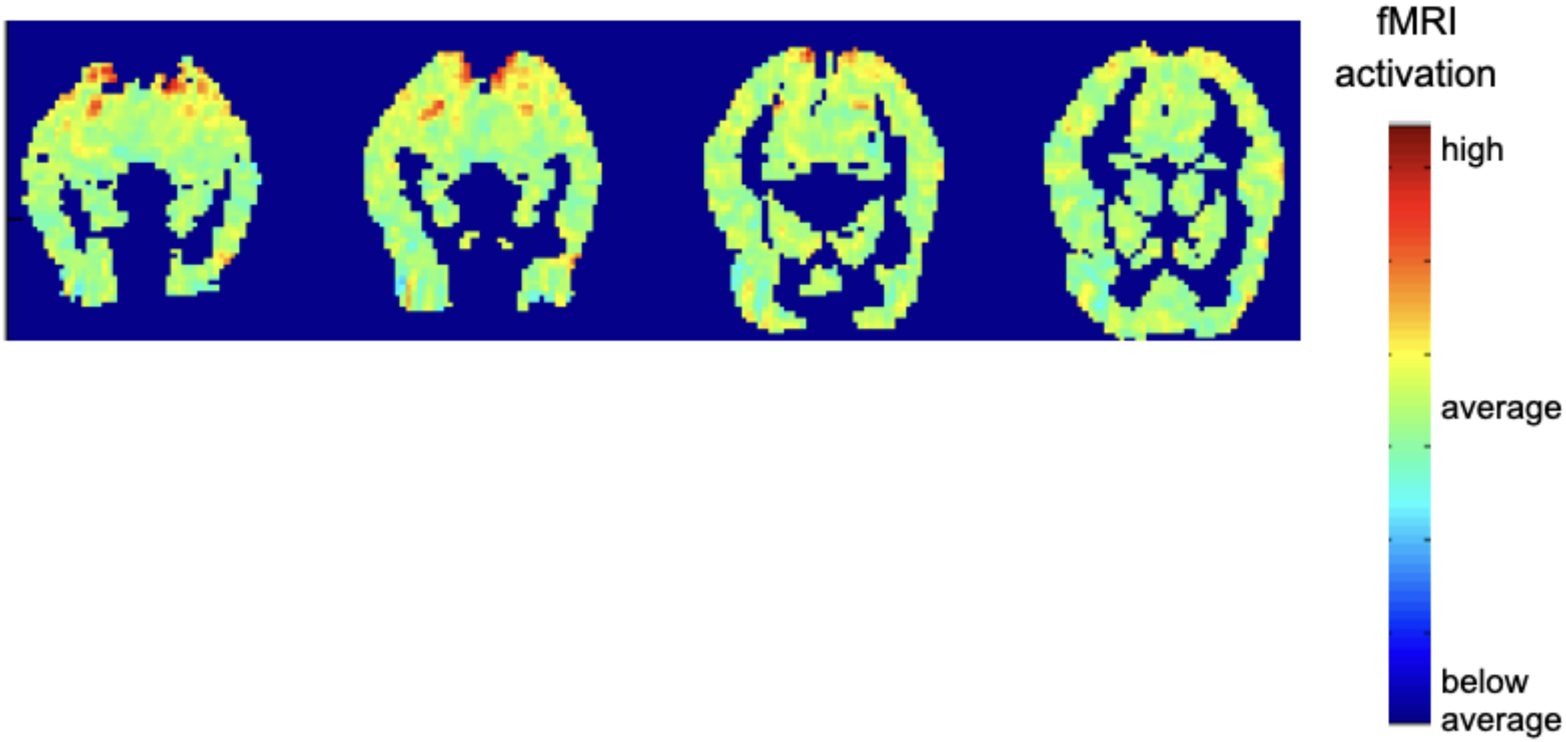
Stimulus for the study



60 distinct exemplars, presented 6 times each

[Mitchell et al. Science 2008 (https://science.sciencemag.org/content/320/5880/1191.abstract)](https://science.sciencemag.org/content/320/5880/1191.abstract), data available [online (https://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html)](https://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html).

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:

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

What can we do?

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

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

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}

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

$$= \underset{y_k}{\operatorname{argmax}}\ \pi_k \prod_i \mathcal{N}(X_i^{\text{new}}; \mu_{ik}, \sigma_{ik})$$

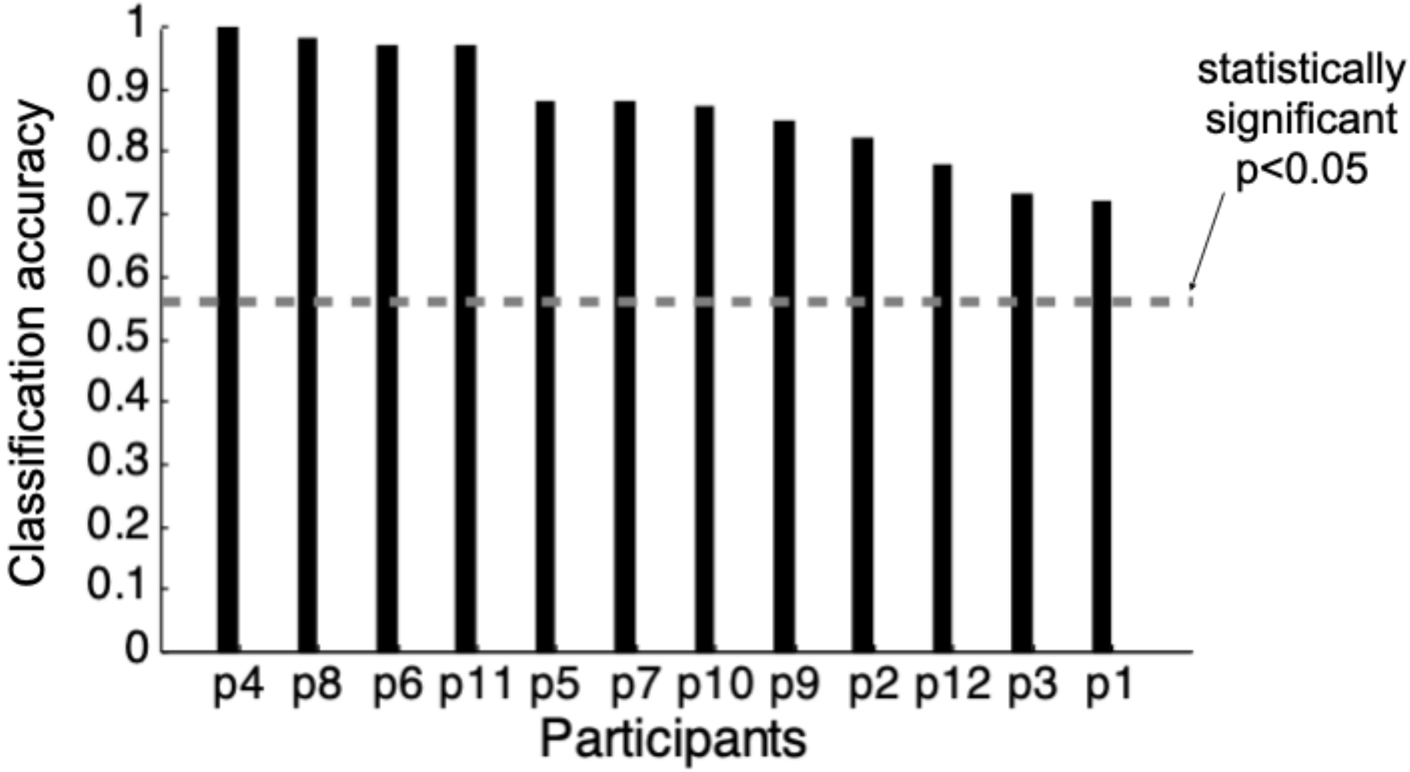
Estimating Parameters: Y discrete, X_i continuous

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

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

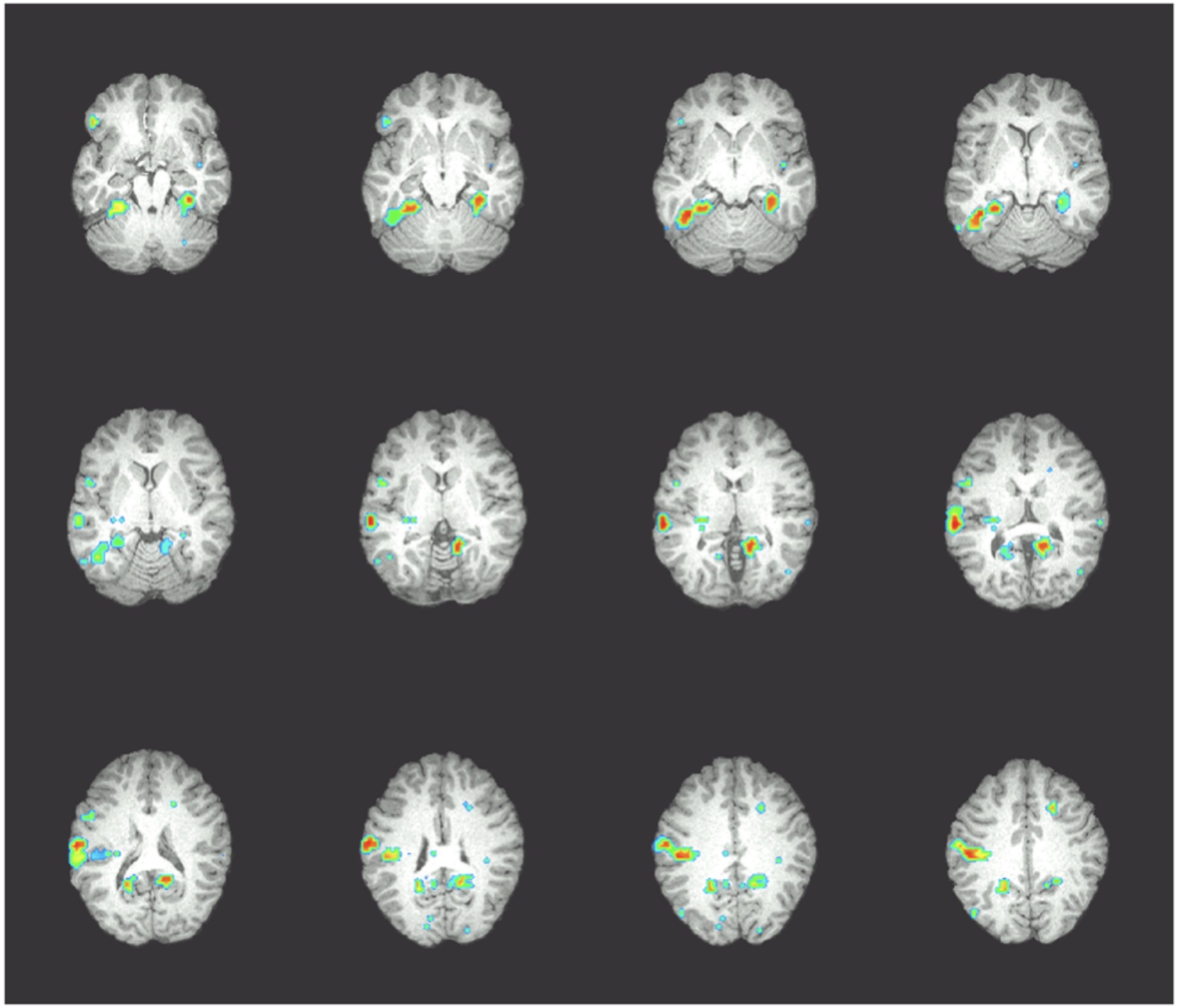
$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

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]

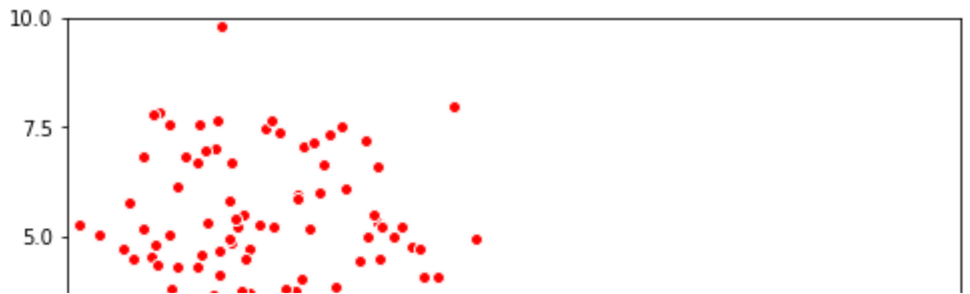


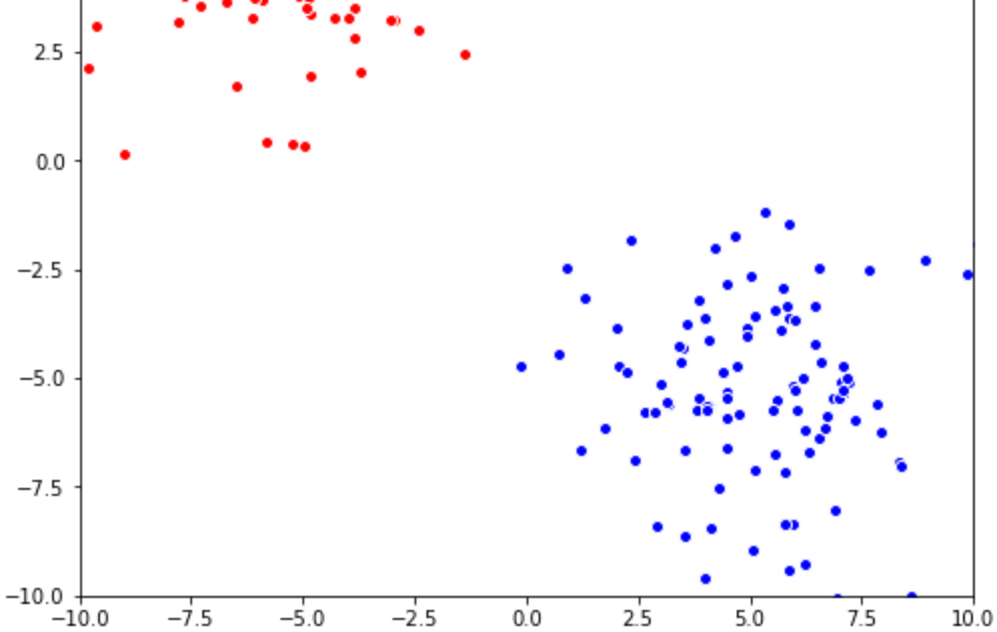
In [15]:

```
import numpy as np
import matplotlib.pyplot as plt
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 = (1
00,2))
X_negative = norm.rvs(loc=[mu_1_0,mu_2_0], scale=[sigma_1_0, sigma_2_0], size = (1
00,2))


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')
plt.axis([-10,10,-10,10],'equal')
```

Out[15]:

```
[-10, 10, -10, 10]
```

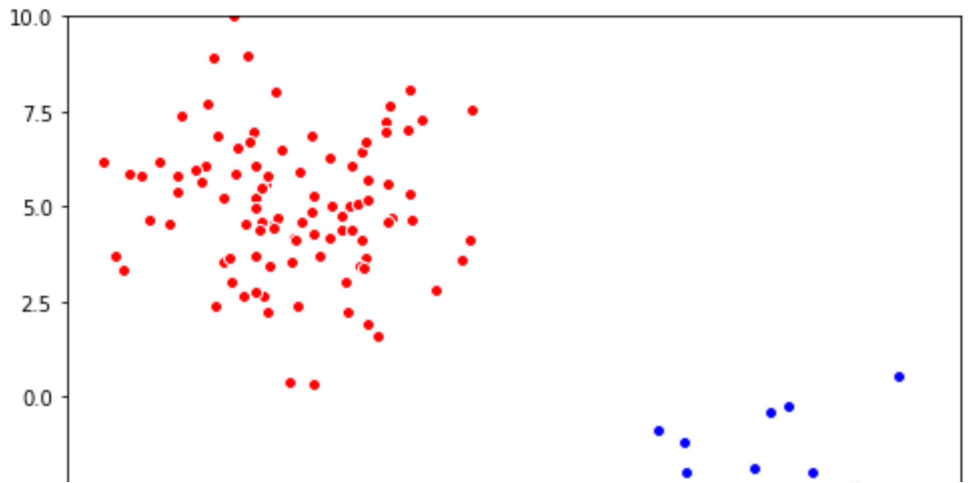


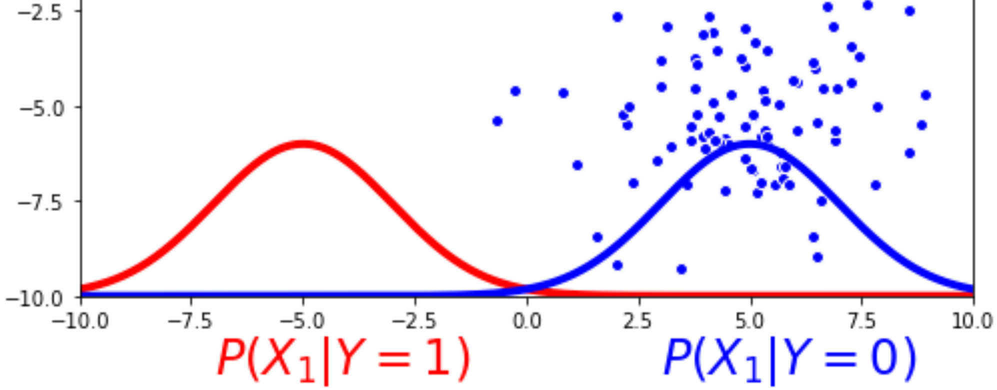


2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

In [2]:
```
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=(8,7))

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=24)

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=24)


plt.axis([-lim_plot,lim_plot,-lim_plot,lim_plot],'equal')
```

Out[2]:
```
[-10, 10, -10, 10]
```

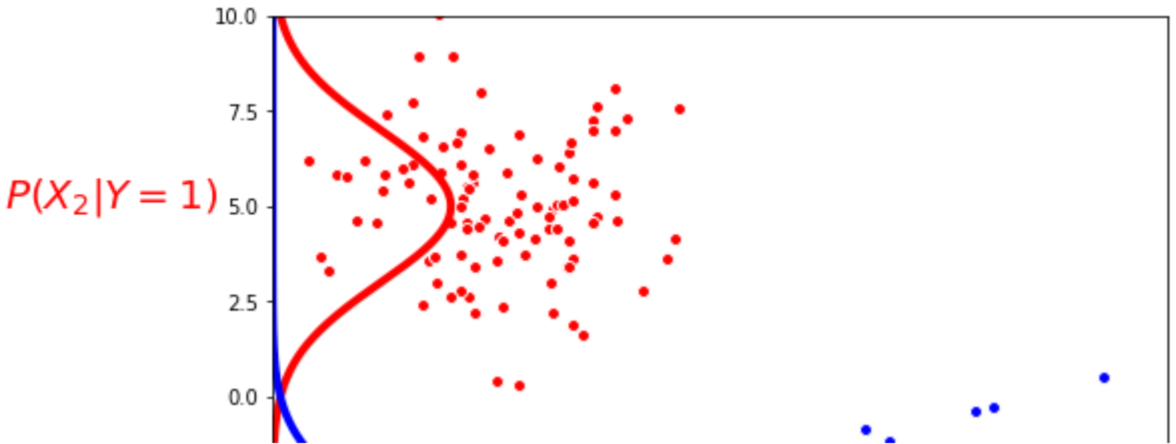


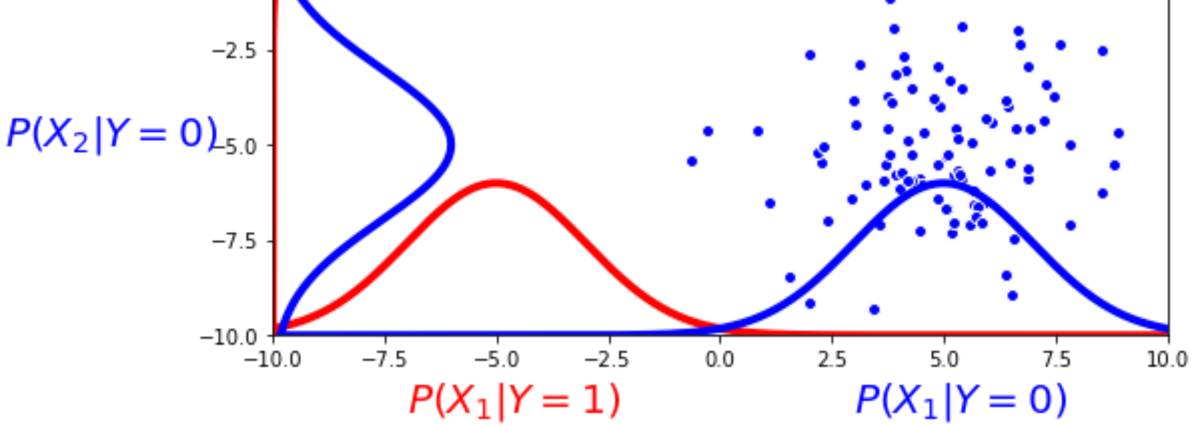


−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
$P(X_1|Y=1)$
$P(X_1|Y=0)$

In [3]:
```python
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(-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],'equal')
```

Out[3]: [-10, 10, -10, 10]





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

In [4]:

```
# 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)
```

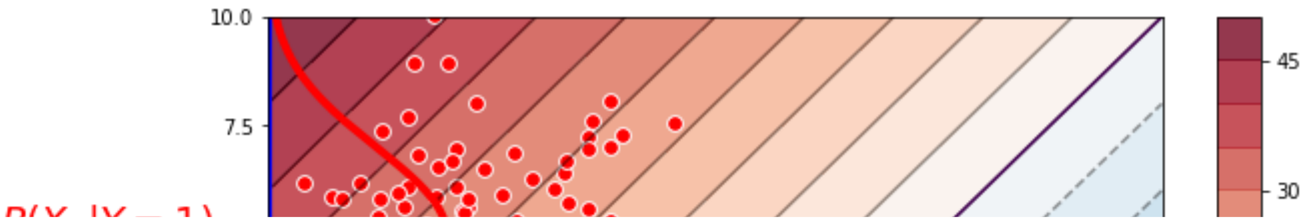
In [5]:

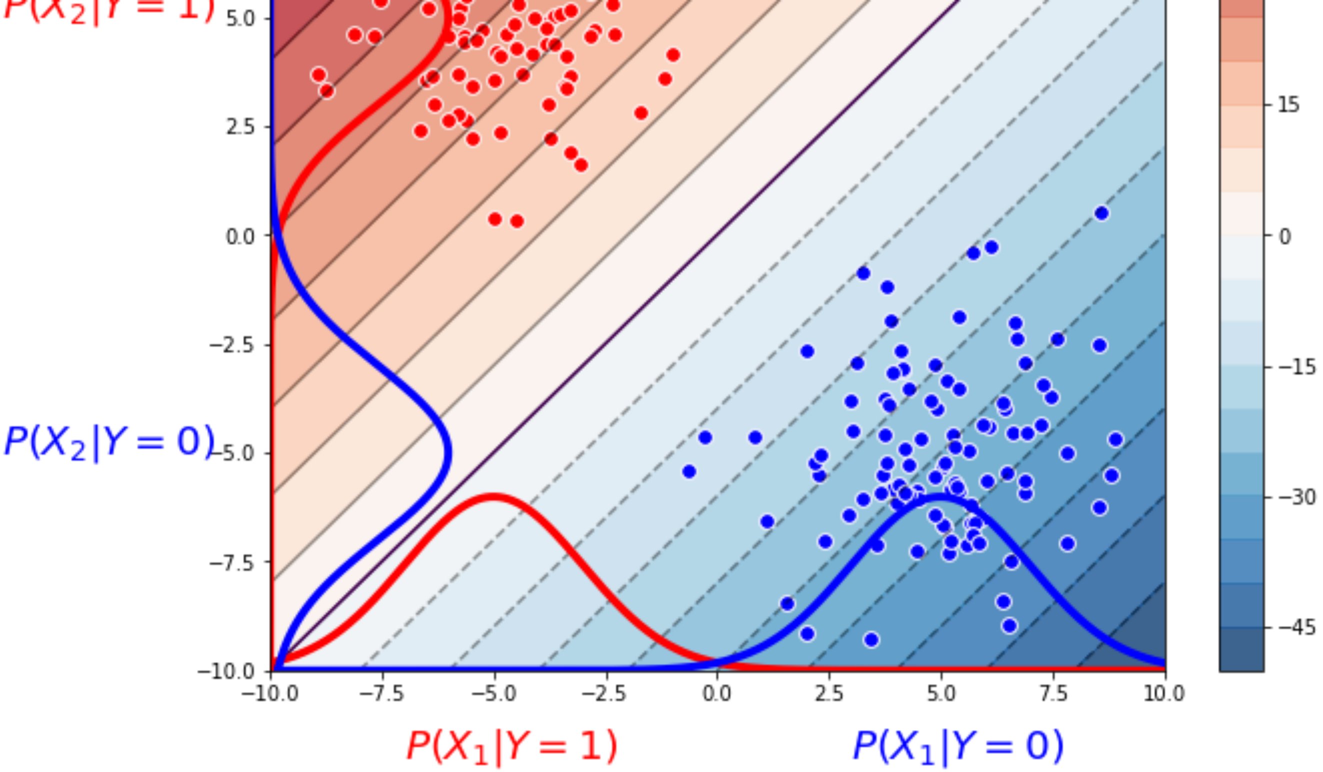
```
plt.figure(figsize=(10,8))

# 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)
plt.contour(contours,levels=[0],linewidth=5)

# 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],'equal')
```

```
/Users/lwehbe/env/py3/lib/python3.7/site-packages/matplotlib/contour.py:1000:
UserWarning: The following kwargs were not used by contour: 'linewidth'
  s)
```

Out[5]:

```
[-10, 10, -10, 10]
```





$P(X_2|Y=1)$
$P(X_2|Y=0)$
$P(X_1|Y=1)$
$P(X_1|Y=0)$
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
15
0
−15
−30
−45

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 save variance

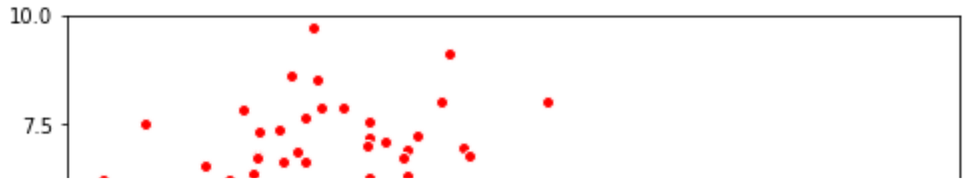
In [6]:

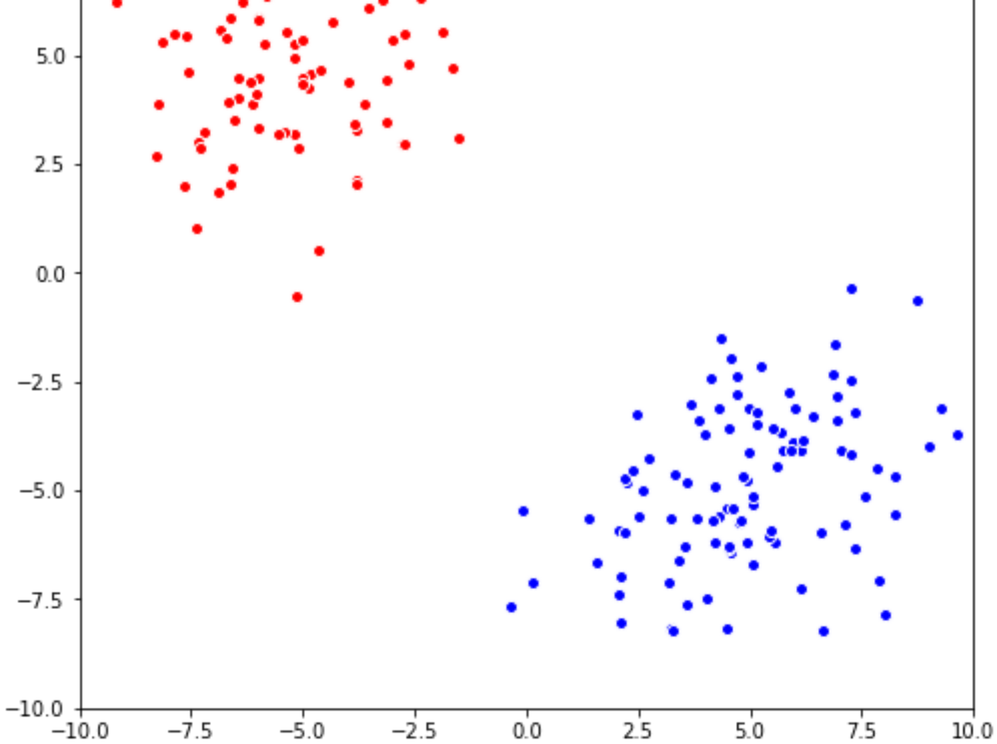
```python
from scipy.stats import multivariate_normal

# 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**2,1.5], [1.5,sigma_2_1**2]] )
cov_negative = np.array([[sigma_1_0**2,1.5], [1.5,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))


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')
plt.axis([-10,10,-10,10],'equal')
```

```
[[4.  1.5]
 [1.5 4. ]]
```

Out[6]:

```
[-10, 10, -10, 10]
```





5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

In [7]:

```python
# 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 , sigma_2 = np.std(X_positive,axis=0)
sigma_1_1, sigma_2_1 = sigma_1 , sigma_2
sigma_1_0, sigma_2_0 = sigma_1 , sigma_2

print(mu_1_1, mu_2_1,mu_1_0, mu_2_0 , sigma_1 , sigma_2)

# 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 estimated parameters
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)

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)
```

```
-5.085426203615889 5.165209290757402 4.889542628155099 -4.881977255527493 2.15
19057229498606 2.0604633328718243
```

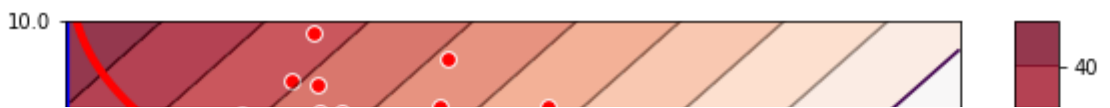
In [8]:

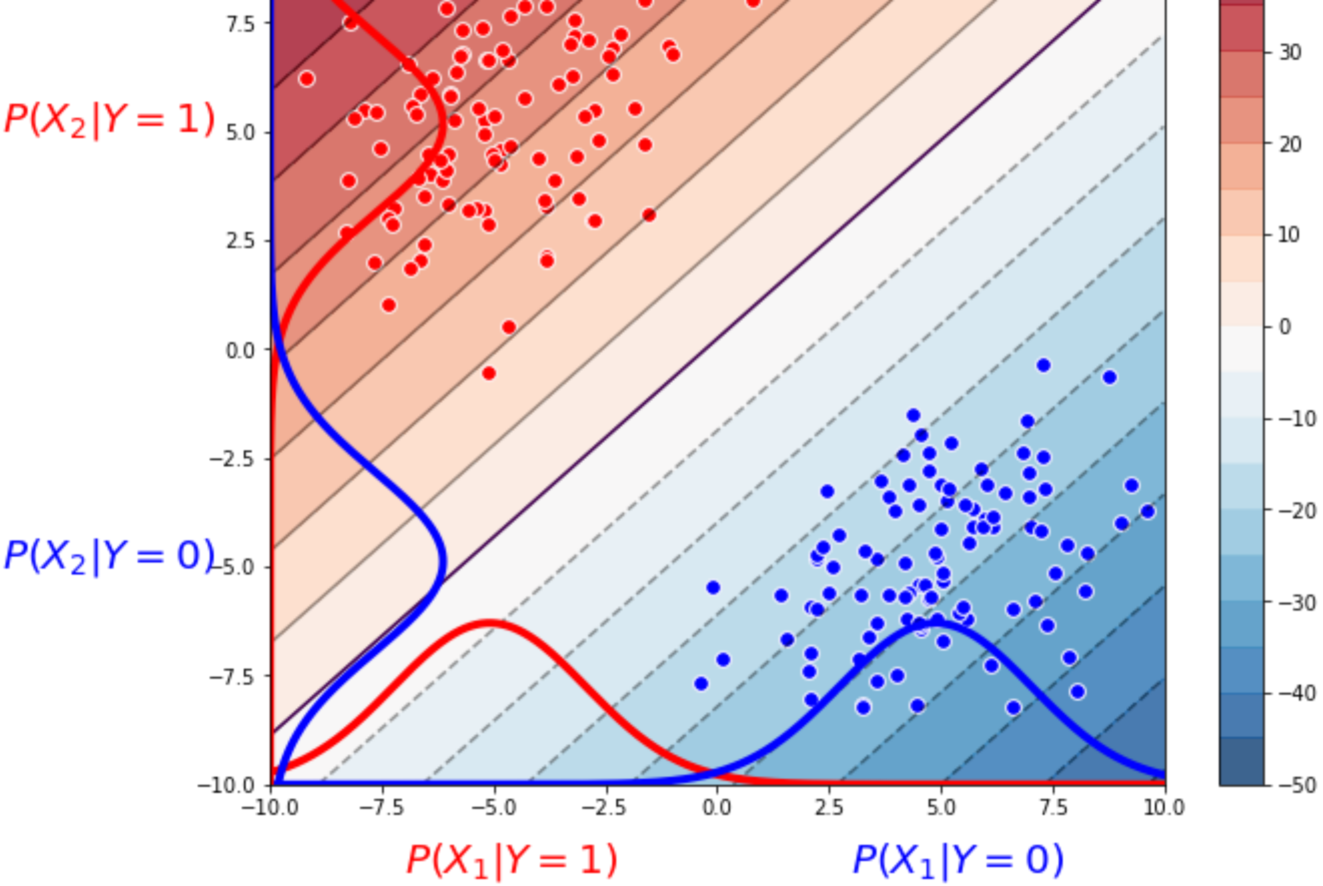
```python
# other things don't change!
print(X_positive.std(axis=0))
print(X_negative.std(axis=0))

plt.figure(figsize=(10,8))

# 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)
plt.contour(contours,levels=[0],linewidth=5)

# 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],'equal')
```

```
[2.15190572 2.06046333]
[2.04247006 1.78256668]
```

Out[8]:

```
[-10, 10, -10, 10]
```





$P(X_2|Y=1)$
$P(X_2|Y=0)$
$P(X_1|Y=1)$
$P(X_1|Y=0)$

Is GNB a linear separator?

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

Decision rule:

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

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

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

Is GNB a linear separator?

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

Decision rule:

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

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

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

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

$$= \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{\frac{1}{\sigma_{i1}}}{\frac{1}{\sigma_{i0}}} \exp\Big(- \frac{1}{2}\frac{(x_i - \mu_{i1})^2}{\sigma_{i1}^2} + \frac{(x_i - \mu}{\sigma_{i0}^2}$$

Is GNB a linear separator?

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

Decision rule:

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

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

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

$$\begin{aligned}
\ln \frac{P(Y=1|X_1 \dots X_d)}{P(Y=0|X_1 \dots X_d)} &= \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{P(X_i|Y=1)}{P(X_i|Y=0)} \\
&= \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{\frac{1}{\sigma_{i1}}}{\frac{1}{\sigma_{i0}}} \exp\Big(- \frac{1}{2}\frac{(x_i - \mu_{i1})^2}{\sigma_{i1}^2} + \frac{(x_i - \mu}{\sigma_{i0}^2} \\
&= \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{\sigma_{i0}}{\sigma_{i1}} + \sum_i \Big(- \frac{1}{2}\frac{(x_i - \mu_{i1})^2}{\sigma_{i1}^2} + \frac{(x_i}{}
\end{aligned}$$

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

Is GNB a linear separator?

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

Decision rule:

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

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

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

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

$$= \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{\frac{1}{\sigma_{i1}}}{\frac{1}{\sigma_{i0}}} \exp\Big(-\frac{1}{2}\frac{(x_i - \mu_{i1})^2}{\sigma_{i1}^2} + \frac{(x_i - \mu}{\sigma_{i0}^2}$$

$$= \ln \frac{P(Y=1)}{P(Y=0)} + \sum_i \ln \frac{\sigma_{i0}}{\sigma_{i1}} + \sum_i \Big(-\frac{1}{2}\frac{(x_i - \mu_{i1})^2}{\sigma_{i1}^2} + \frac{(x_i}{}$$

What happens if we force $\hat{\sigma}_{i0} = \hat{\sigma}_{i1}$?

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

In [9]:

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

# 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 estimated parameters
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)

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)
```

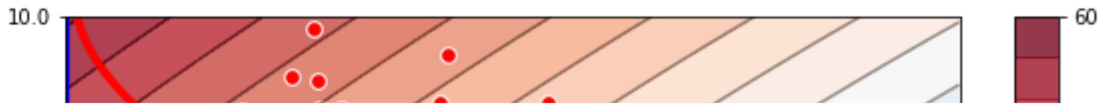
In [10]:

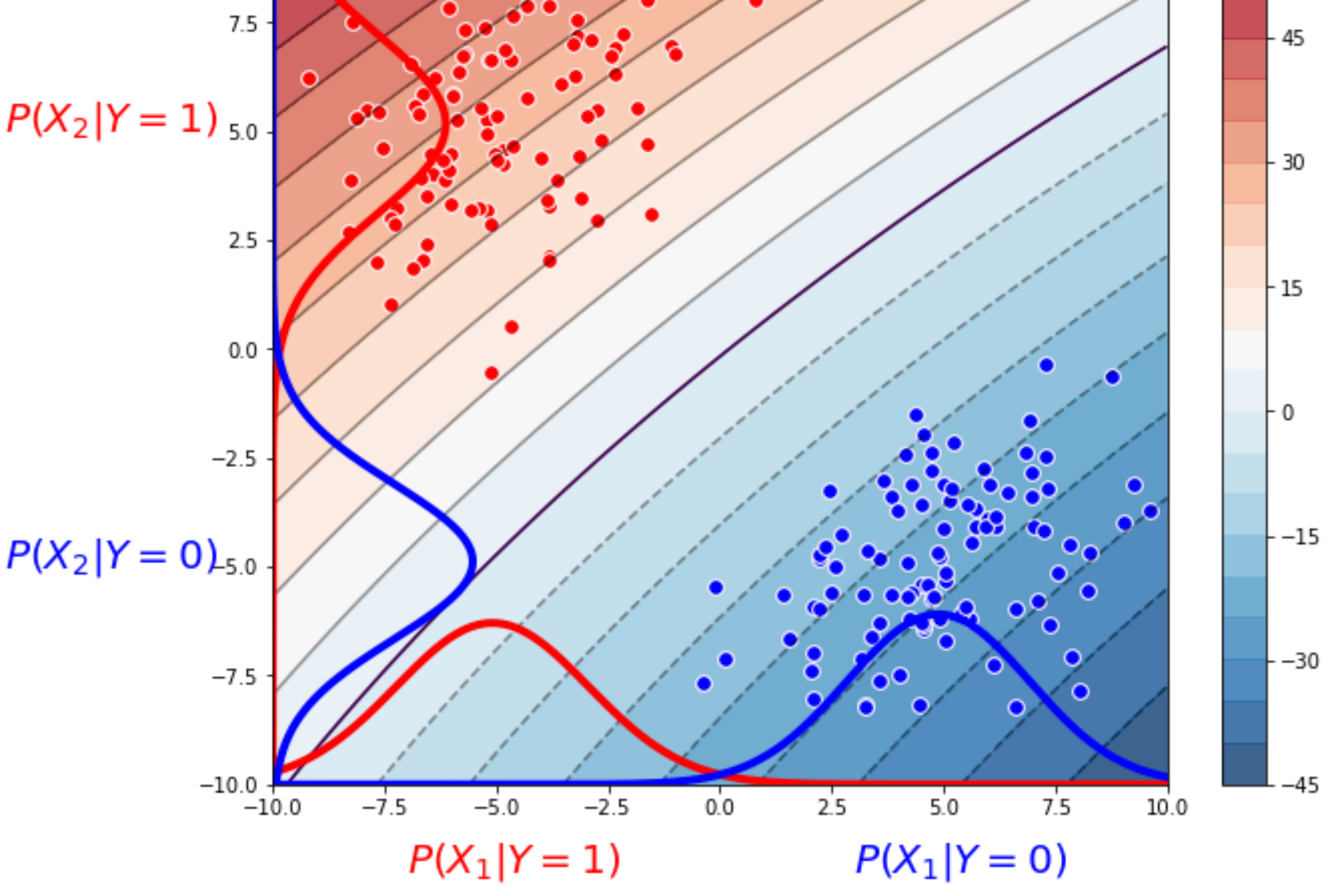
```python
# other things don't change!
print(X_positive.std(axis=0))
print(X_negative.std(axis=0))

plt.figure(figsize=(10,8))

# 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)
plt.contour(contours,levels=[0],linewidth=5)

# 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],'equal')
```

```
[2.15190572 2.06046333]
[2.04247006 1.78256668]
```

Out[10]:

```
[-10, 10, -10, 10]
```





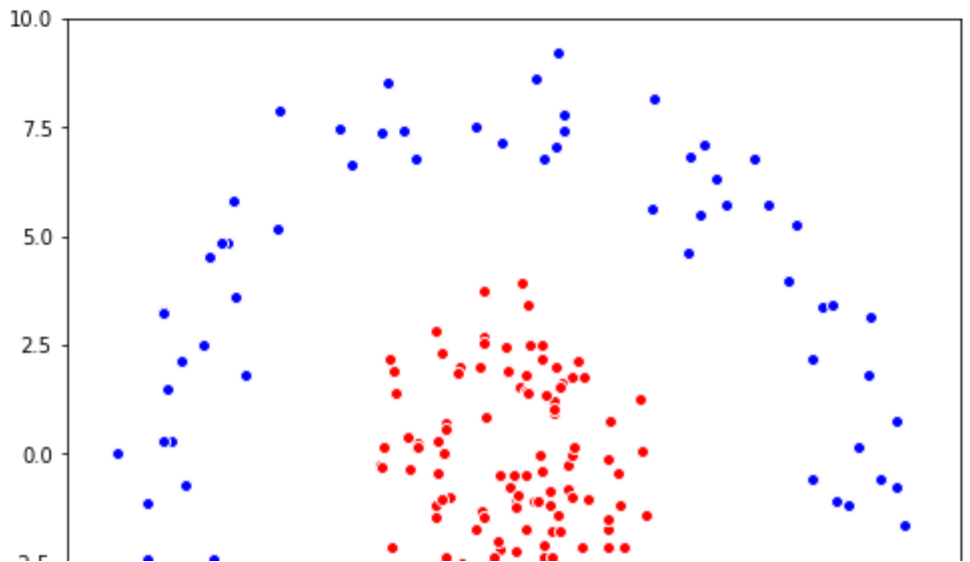
$P(X_2|Y=1)$
$P(X_2|Y=0)$
$P(X_1|Y=1)$
$P(X_1|Y=0)$

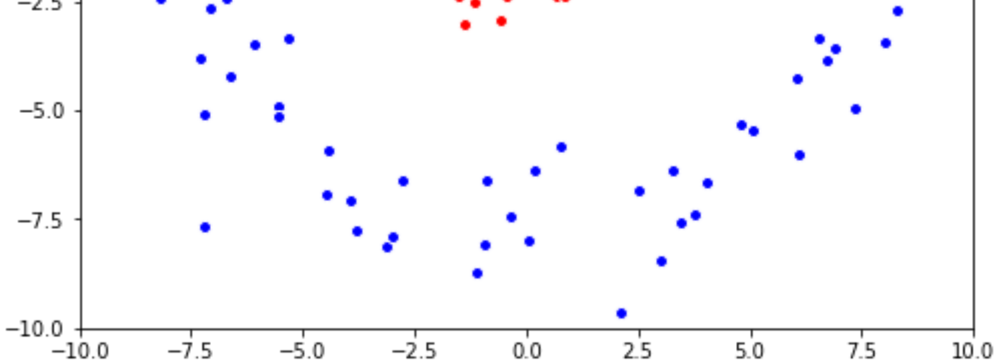
In [19]:

```
from sklearn import datasets

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


X_positive = X[y==1]*8
X_negative = X[y==0]*8


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')
plt.axis([-10,10,-10,10],'equal')
```

Out[19]:

```
[-10, 10, -10, 10]

<Figure size 360x360 with 0 Axes>
```





−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

In [20]:

```python
# Estimate mean
mu_1_1, mu_2_1 = np.mean(X_positive,axis=0)
mu_1_0, mu_2_0 = np.mean(X_negative,axis=0)

# # Estimate different std
# sigma_1_1, sigma_2_1 = np.std(X_positive,axis=0)
# sigma_1_0, sigma_2_0 = np.std(X_negative,axis=0)

# # Estimate same std
sigma_1 , sigma_2 = np.std(X_positive,axis=0)
sigma_1_1, sigma_2_1 = sigma_1 , sigma_2
sigma_1_0, sigma_2_0 = sigma_1 , sigma_2


# 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 estimated parameters
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)

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)
```

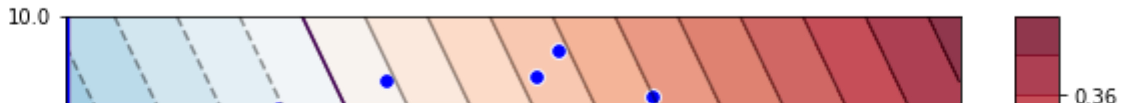
In [21]:

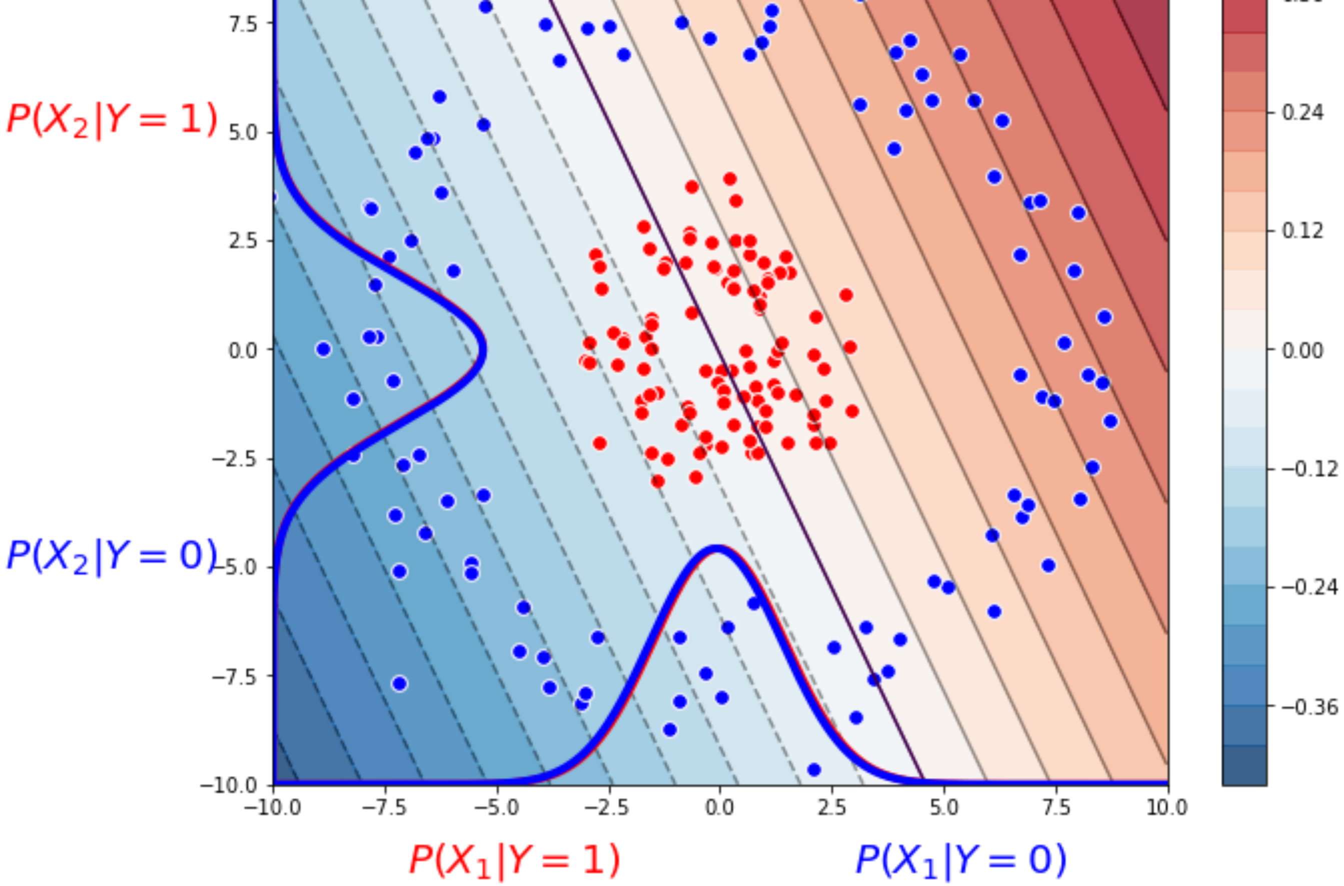
```python
# other things don't change!
print(X_positive.std(axis=0))
print(X_negative.std(axis=0))

plt.figure(figsize=(10,8))

# 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)
plt.contour(contours,levels=[0],linewidth=5)

# 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],'equal')
```

```
[1.47226964 1.69535998]
[5.6735921 5.5715988]
```

Out[21]:

```
[-10, 10, -10, 10]
```





$P(X_2|Y=1)$
$P(X_2|Y=0)$
$P(X_1|Y=1)$
$P(X_1|Y=0)$
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
0.24
0.12
0.00
−0.12
−0.24
−0.36