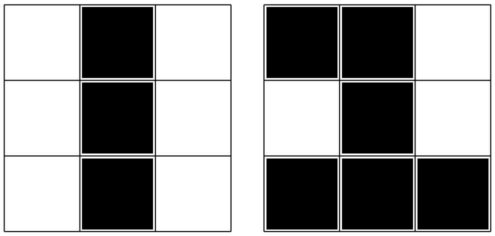
The following shows a worked example of a simple Naive Bayes classifier with 3 training data images each only 3x3 pixels. Images are classified as either a 0 or a 1. For simplicity, this example only works with black and white pixel values.

This example assumes you have read the handout first and is meant to be supplementary to the handout. Certain things, like probability notation, will not be explained here.

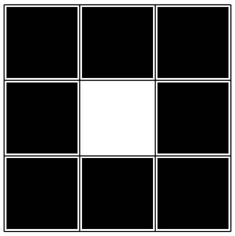
If you think you are already quite familiar with the theory behind Naive Bayes classification, you can skip ahead to the section labeled "The Example" on page 3

1 The Setup

The 2 training images in the training data with classification 1:



The only training image in the training data with classification 0:



For the purposes of this very small example, that is all the training data we will work with.

With all the image data having been declared, let's start with Bayes' Theorem:

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

In our case, we are interested in the probability that the class of an image is c given all the pixels in that image. That is:

$$P(class = c|\text{all pixel values}) = \frac{P(\text{all pixel values}|class = c) * P(class = c)}{P(\text{all pixel values})}$$

For our purposes, actually computing $P(\text{all pixel values})$ in the denominator is entirely irrelevant because this probability will be the same for all classes. At the risk of oversimplifying things: every probability computed for the image we are classifying will be divided by the same denominator, so the denominator can essentially be factored out. Therefore what we actually care about is the following proportion:

$$P(class = c|\text{all pixel values}) \propto P(class = c) * P(\text{all pixel values}|class = c)$$

Estimating $P(class = c)$ for our implementation of Naive Bayes is done by using the proportion of training images that belong to class c compared to the total amount of training images. As stated in the handout:

$$P(class = c) = \frac{\text{\# of training examples where class = c}}{\text{\# of training examples}}$$

Now all we need to do is figure out how to obtain $P(\text{all pixel values}|class = c)$. To do this we need to expand on the definition of that probability, what it means to find a probability for all pixel values in the image being classified when given a class. In this implementation of Naive Bayes, our only features are the pixel values. We therefore have one feature we can use in classification for every pixel in the image, so 9 total features in this example. Graphically, our features are the following:

$F_{0,0}$	$F_{0,1}$	$F_{0,2}$
$F_{1,0}$	$F_{1,1}$	$F_{1,2}$
$F_{2,0}$	$F_{2,1}$	$F_{2,2}$

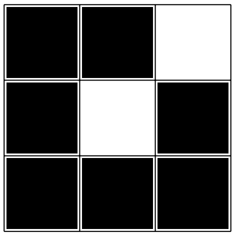
Let lowercase $f_{i,j}$ correspond to the value of $F_{i,j}$ in the image that is currently being classified. That is, $f_{i,j}$ is shorthand for $(F_{i,j} = value)$. Using this, we can now expand on the notation from before:

$$P(\text{all pixel values}|class = c) = P(f_{0,0}|class = c)*P(f_{0,1}|class = c)*...*P(f_{2,2}|class = c)$$

We can now directly compute $P(f_{i,j}|class = c)$ for every value of i and j and every possible class c from what we observed in the training data. Having done this setup work and notation defining, let's actually do the example:

2 The Example

Consider the following example image that we wish to classify:



This image we are trying to classify is clearly closer to a 0 than it is a 1, so we would expect the final output classification to be a 0.

First, we have a massive problem we need to deal with.
$P(F_{0,2} = white|class = 0)$ is 0 and $P(F_{1,0} = black|class = 1)$ is 0. We have absolutely never seen training data with all the current image pixel values, so all probabilities will come out to 0. This is why we use Laplace Smoothing, it adds at least some way to account for data that was never seen in the training set. As stated in the handout, we will now instead compute:

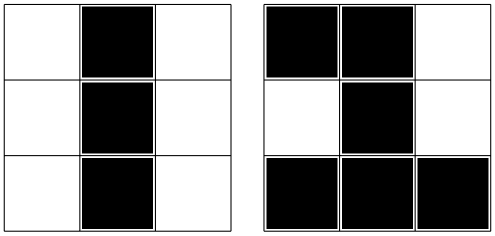
$$P(F_{i,j} = f|class = c) = \frac{k + \#\text{ of times } F_{i,j} = f \text{ when class = c}}{2k + \text{Total number of training examples where class = c}}$$

Let $k = 1$ for our example. This value was chosen arbitrarily. Higher or lower values of k may perform much better for you.

Now to actually apply the Naive Bayes algorithm:
We will start with class 1 in this example because it has more training data. It will be more clear for demonstration purposes. This class has 2 images in the training data out of 3 total training images.

$$P(class = 1) = 2/3$$
$$\log(P(class = 1)) = -0.176$$

As a reminder this is all the training data for class 1:



The top left pixel is black in our image being classified. This corresponds to $F_{0,0} = black$. In our training data for class 1 shown above, $F_{0,0}$ was black 1 time out of 2 training images, therefore

$$P(F_{0,0} = black|class = 1) = 1/2$$

We use the training data to compute the probability that the current feature would have the value it does if the image we are classifying actually belongs to class 1. This process is repeated for every feature value in the image we are classifying with the assumption the image has class 1. This gives the following table:

	$P(F_{i,j} = f_{i,j}\|class = 1)$	Smoothed	log(Smoothed)
$F_{0,0} = black$	1/2	2/4	-0.301
$F_{0,1} = black$	2/2	3/4	-0.125
$F_{0,2} = white$	2/2	3/4	-0.125
$F_{1,0} = black$	0/2	1/4	-0.602
$F_{1,1} = white$	0/2	1/4	-0.602
$F_{1,2} = black$	0/2	1/4	-0.602
$F_{2,0} = black$	1/2	2/4	-0.301
$F_{2,1} = black$	2/2	3/4	-0.125
$F_{2,2} = black$	1/2	2/4	-0.301

Now we can compute

$$P(class = 1|\text{all pixel values}) \propto P(class = 1) * P(\text{all pixel values}|class = 1)$$

$$P(class = 1|\text{all pixel values}) \propto (\frac{2}{3})*((\frac{2}{4})(\frac{3}{4})(\frac{3}{4})(\frac{1}{4})(\frac{1}{4})(\frac{1}{4})(\frac{2}{4})(\frac{3}{4})(\frac{2}{4})) \approx 0.00055$$

You can see how even for a 3x3 example these proportions are extremely small. On the full 28x28 images in the assignment, this would quickly become impossible to work with. This is why we use logarithms of probabilities instead:

$$\log(P(class = 1|\text{all pixel values})) \propto \log(P(class = 1)*P(\text{all pixel values}|class = 1))$$

Using that $\log(a * b) = \log(a) + \log(b)$, this becomes

$$\log(P(class = 1|\text{all pixel values})) \propto \log(P(class = 1))+\log(P(\text{all pixel values}|class = 1))$$

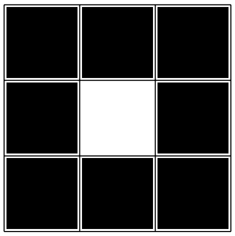
$$\log(P(class = 1|\text{all pixel values})) \propto -0.176+(-0.301+-0.125+-0.125+...+-0.301) \approx -3.26$$

And that is our final proportion value for if this image belongs to class 1.

We now move on to class 0. This class has 1 image in the training data out of 3 total training images:

$$P(class = 0) = 1/3$$
$$\log(P(class = 0)) = -0.477$$

As a reminder, this is all training data for class 0:



	$P(F_{i,j} = f_{i,j} \vert class = 0)$	Smoothed	log(Smoothed)
$F_{0,0} = black$	1/1	2/3	-0.176
$F_{0,1} = black$	1/1	2/3	-0.176
$F_{0,2} = white$	0/1	1/3	-0.477
$F_{1,0} = black$	1/1	2/3	-0.176
$F_{1,1} = white$	1/1	2/3	-0.176
$F_{1,2} = black$	1/1	2/3	-0.176
$F_{2,0} = black$	1/1	2/3	-0.176
$F_{2,1} = black$	1/1	2/3	-0.176
$F_{2,2} = black$	1/1	2/3	-0.176

$$P(class = 0|\text{all pixel values}) \propto (\frac{1}{3}) * ((\frac{2}{3})(\frac{2}{3})(\frac{1}{3})(\frac{2}{3})(\frac{2}{3})(\frac{2}{3})(\frac{2}{3})(\frac{2}{3})(\frac{2}{3})) \approx 0.004$$

$$\log(P(class = 0|\text{all pixel values})) \propto -0.477+(-0.176+-0.176+-0.477+...+-0.176) \approx -2.362$$

We have now computed the proportion values for all classes. All that is left to do is compare them and determine how we should classify the current image. The magnitude of these proportion values does not matter. All we care about right now is which class has the largest proportion value. The image being classified is most similar to this class under our current model.

	$P(class = c\vert\text{all pixel values}) \propto$	$\log(P(class = c\vert\text{all pixel values})) \propto$
$class = 0$	0.004	-2.362
$class = 1$	0.00055	-3.26

-2.362 is the highest log proportion value for all possible image classes, therefore we correctly conclude that the image we are classifying has $class = 0$.

One last note is that I have been careful not to call the final output of the Naive Bayes algorithm probabilities. Neither 0.004 nor -2.362 are actually probabilities and as such you should NOT expect these values to follow the laws of probability.