Naïve Bayes Classification/Gaussian Discriminant Analysis

Kazi Aminul Islam
Department of Computer Science
Kennesaw State University

Andrew Ng, Stanford University, Naïve Bayes Lecture Notes.

Material borrowed from Jonathan Huang and I. H. Witten's and E. Frank's "Data Mining" and Jeremy Wyatt and others

Things We'd Like to Do

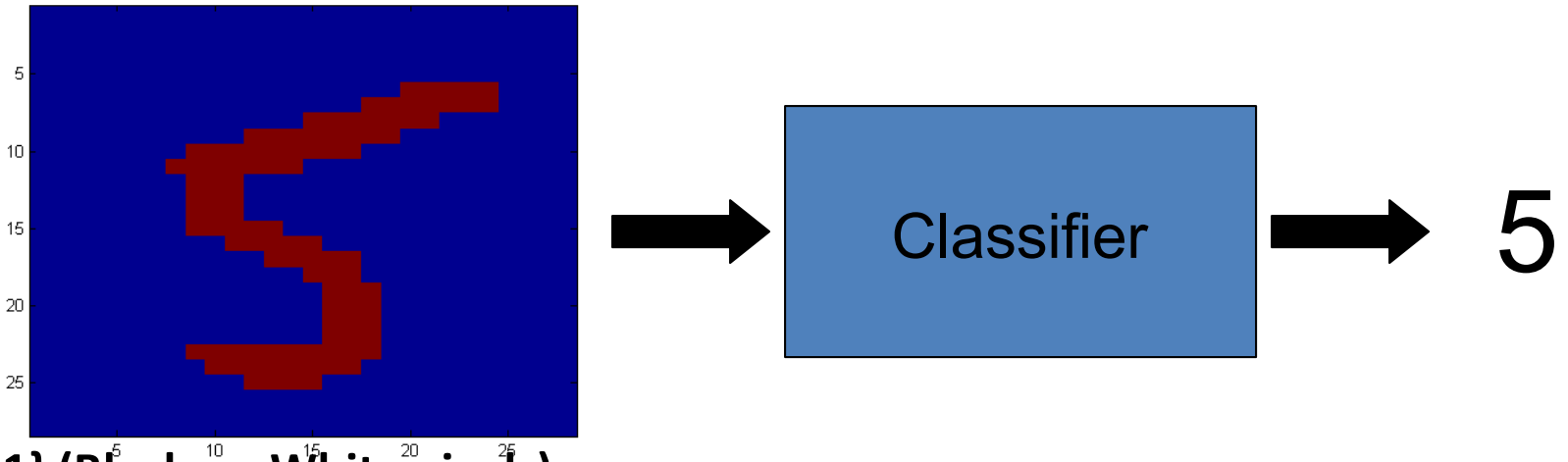
- Spam Classification
  - Given an email, predict whether it is spam or not
- Medical Diagnosis
  - Given a list of symptoms, predict whether a patient has disease X or not
- Weather
  - Based on temperature, humidity, etc… predict if it will rain tomorrow

Bayesian Classification

- Problem statement:
  - Given features $X_1, X_2, \ldots, X_n$
  - Predict a label Y
- Generative learning
  - Given x, we are learning p(x|y) and p(y) to predict p(y|x).
  - Bayesian classification
- Discriminative learning
  - Given x, we are learning p(y|x) to predict H(p(y|x))=+1 or -1
  - SVM

Another Application

- **Digit Recognition**



- **$X_1,\ldots,X_n \in \{0,1\}$ (Black vs. White pixels)**
- **$Y \in \{5,6\}$ (predict whether a digit is a 5 or a 6)**

The Bayes Classifier

- A good strategy is to predict:

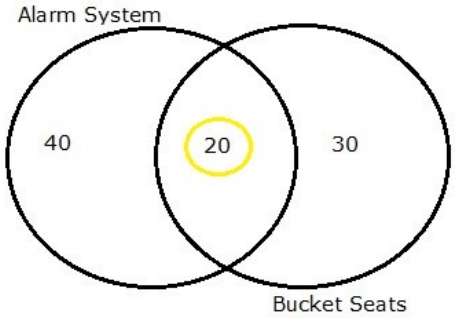
$$\arg\max_Y P(Y|X_1,\ldots,X_n)$$

  - (for example: what is the probability that the image represents a 5 given its pixels?)

- So … How do we compute that?

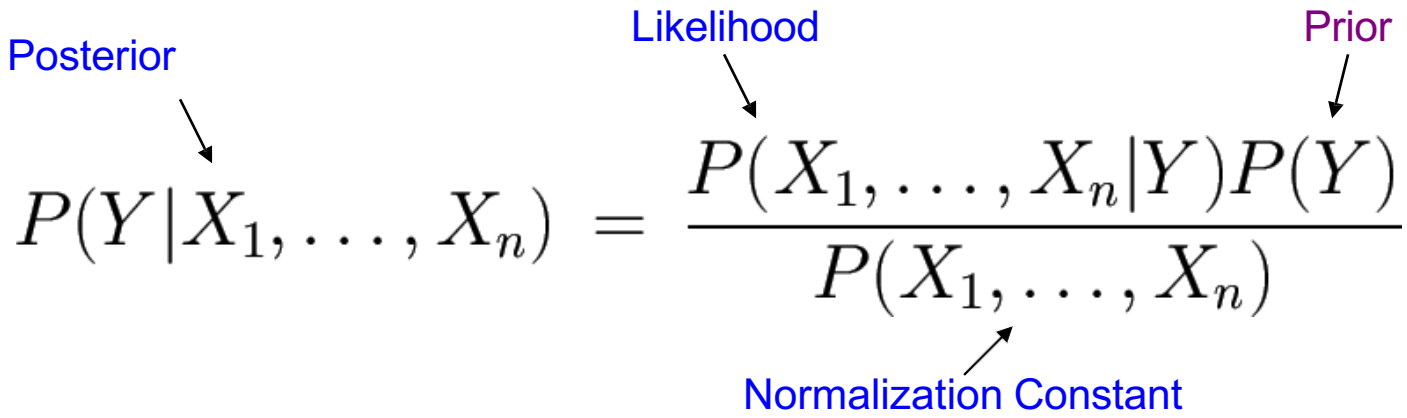
Bayes Rule

- Conditional probability $p(Y|X) = p(Y \cap X)/p(X)$
- Example
  - In a group of 100 sports car buyers, 40 bought alarm systems, 30 purchased bucket seats, and 20 purchased an alarm system and bucket seats. If a car buyer chosen at random bought an alarm system, what is the probability they also bought bucket seats?
  - $p(B|A) = \frac{p(B \cap A)}{p(A)} = \frac{0.2}{0.4} = 0.5$
- Bayes rule $p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$
  - $p(B|A) = \frac{p(B \cap A)}{p(A)} = \frac{0.2}{0.4} = 0.5 = \frac{p(A \cap B)}{p(A)} = \frac{p(A|B)p(B)}{p(A)}$



The Bayes Classifier

- Use Bayes Rule!



- Posterior: probability of the class Y given X
- Likelihood: probability of X belongs to the class Y
- Why did this help? Well, we think that we might be able to specify how features are "generated" by the class label

The Bayes Classifier

- Let's expand this for our digit recognition task:

$$P(Y=5|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=5)P(Y=5)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

$$P(Y=6|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=6)P(Y=6)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

- To classify, we'll simply compute these two probabilities and predict based on which one is greater

Model Parameters

- For the Bayes classifier, we need to “learn” two functions, the likelihood and the prior

- How many parameters are required to specify the prior for our digit recognition example?

Model Parameters

- How many parameters are required to specify the likelihood?
  - (Supposing that each image is 28x28 pixels)

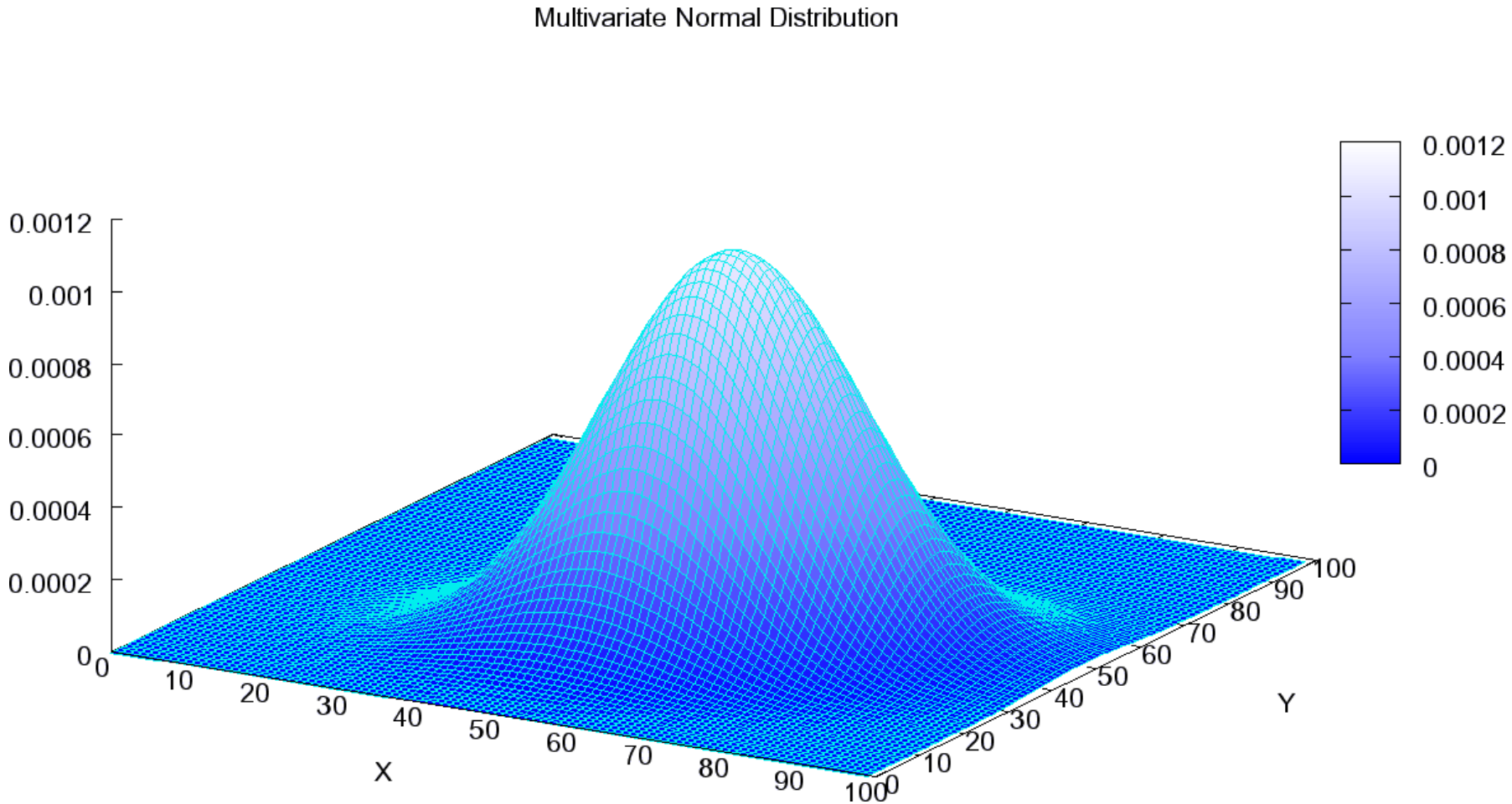
?

Model Parameters

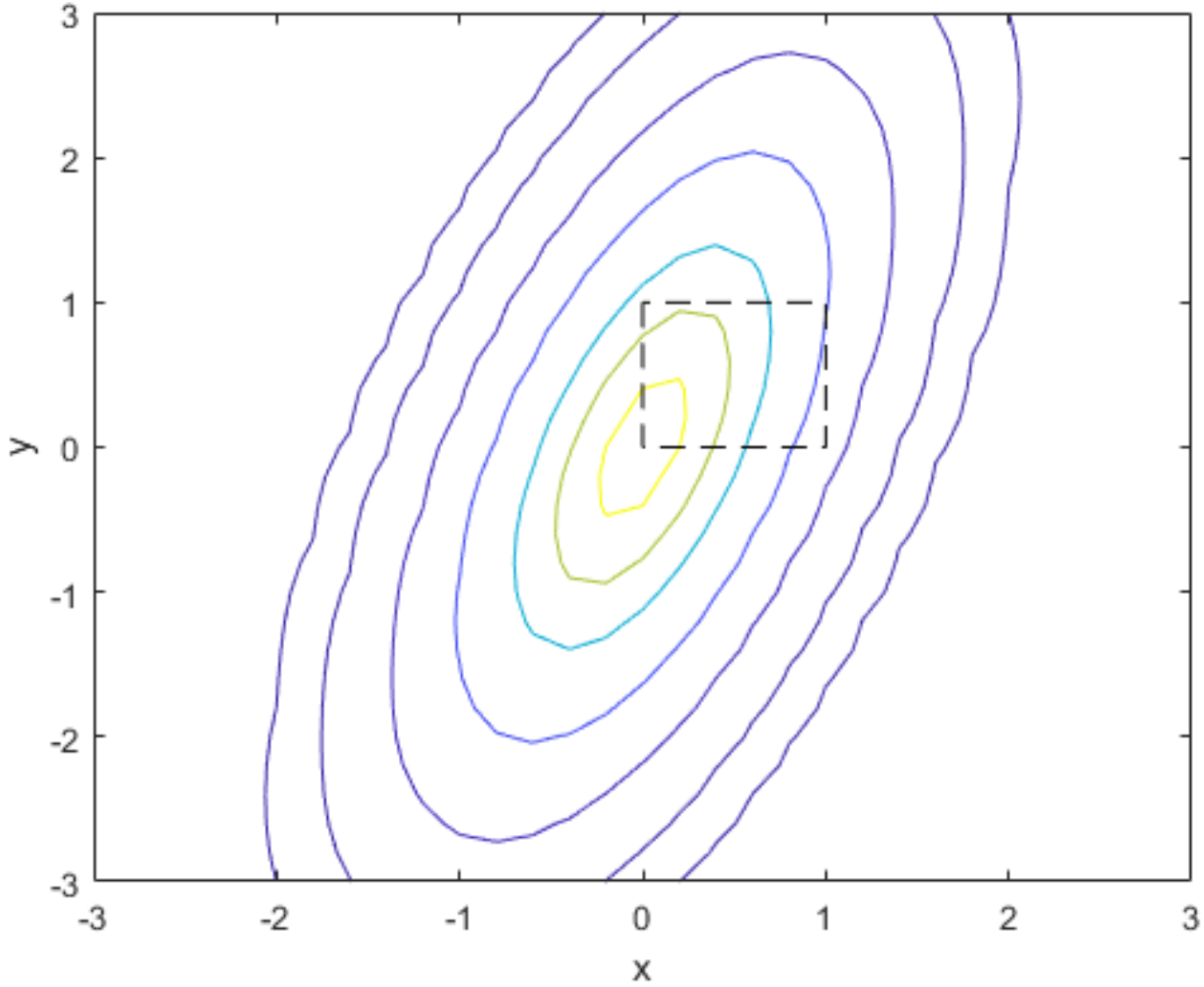
- The problem with explicitly modeling $P(X_1,\ldots,X_n|Y)$ is that there are usually way too many parameters:
  - Each image is of size 28x28, 784 pixels per image. Each pixel can take on integer values in the range of 0..255 inclusive.
  - Model this as a discrete probability distribution, you'd have 255^784 different possibilities.
  - We'll run out of space
  - We'll run out of time
  - And we'll need tons of training data (which is usually not available)
  - So multivariate Gaussian distribution

Gaussian Discriminant Analysis

- Let $x \in \mathbb{R}^n$
- Assume $p(x|y)$ is Gaussian. That is all samples in class y are drawing from a Gaussian distribution.
- For multivariate Gaussian $z \sim N(\mu, \Sigma)$ where $z \in \mathbb{R}^n, \mu \in \mathbb{R}^n, \Sigma \in \mathbb{R}^{n \times n}$
- $E[z] = \mu, \Sigma = \mathrm{Cov(z)} = \mathrm{E}[(\mathrm{z} - \mu)(z - \mu)^T] = E[zz^T] - E[z]E[z]^T$
- $p(z) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$


Multivariate Normal Distribution
0.0012
0.001
0.0008
0.0006
0.0004
0.0002
0
0
10
20
30
40
50
60
70
80
90
100
X
Y

Contour in Multivariate Gaussian Distribution



$$\Sigma = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

GDA Model

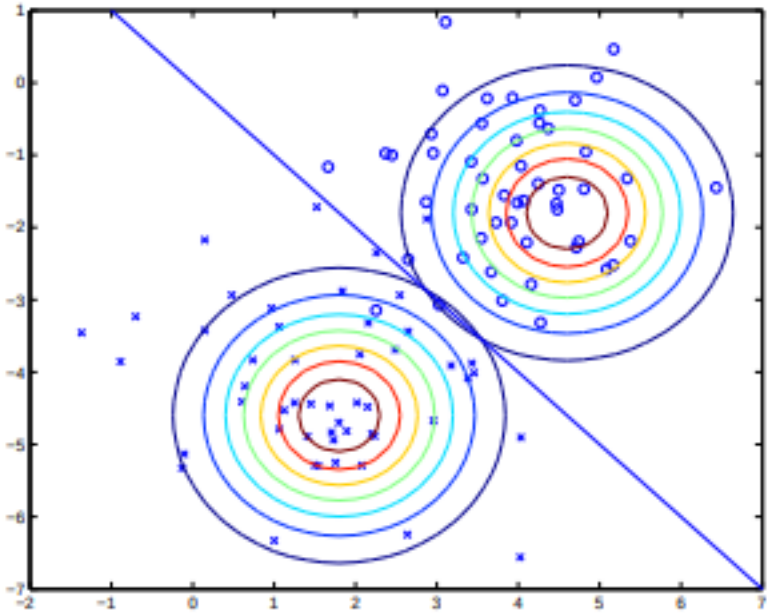
- $p(x|y=0) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)}$
- $p(x|y=1) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}$
- $p(y) = \phi^y(1-\phi)^{1-y}, \mathrm{p(y=1)} = \phi$

- Parameters: $\mu_0, \mu_1 \in R^n, \Sigma \in R^{n\times n}, \phi \in R$

Objective of GDA Model

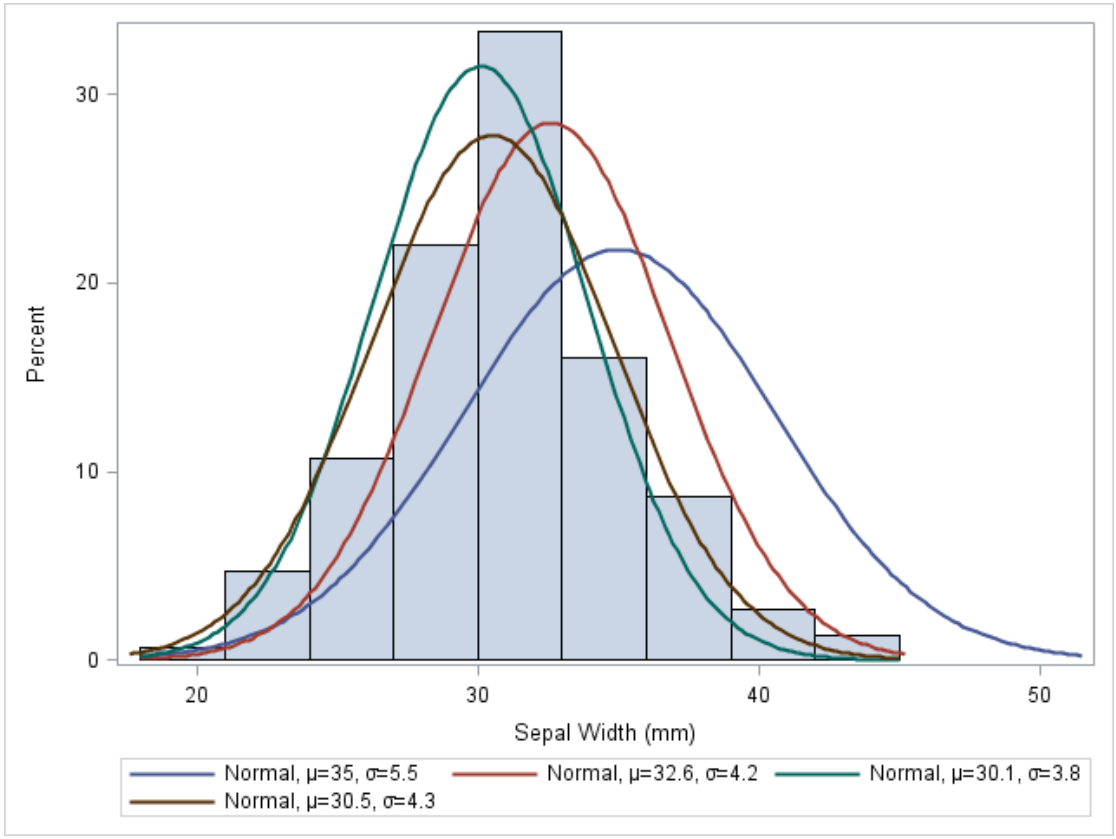
- Given a training set $\{(x_i, y_i)\}_{i=1}^{m}$
- Joint likelihood $\mathcal{L}(\phi, \mu_0, \mu_1, \Sigma) = \Pi_{i=1}^{m} p(x_i, y_i; \phi, \mu_0, \mu_1, \Sigma)$
- $= \Pi_{i=1}^{m} p(x_i | y_i) p(y_i)$

Pictorially, what the algorithm is doing can be seen in as follows:



Maximum Likelihood Estimation (MLE)

- *A method of estimating the parameters of a distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable*



https://towardsdatascience.com/maximum-likelihood-estimation-explained-normal-distribution-6207b322e47f

MLE

- The objective of Maximum Likelihood Estimation is to find the set of parameters (*theta*) that maximize the likelihood function, e.g. result in the largest likelihood value.
  •maximize L(X ; theta)

MLE

Statement of the Problem

Suppose we have a random sample $X_1, X_2, \cdots, X_n$ whose assumed probability distribution depends on some unknown parameter θ. Our primary goal here will be to find a point estimator $u(X_1, X_2, \cdots, X_n)$, such that $u(x_1, x_2, \cdots, x_n)$ is a "good" point estimate of θ, where $x_1, x_2, \cdots, x_n$ are the observed values of the random sample. For example, if we plan to take a random sample $X_1, X_2, \cdots, X_n$ for which the X_i are assumed to be normally distributed with mean μ and variance σ^2, then our goal will be to find a good estimate of μ, say, using the data $x_1, x_2, \cdots, x_n$ that we obtained from our specific random sample.

https://online.stat.psu.edu/stat415/lesson/1/1.2

MLE

suppose we have a random sample $X_1, X_2, \cdots, X_n$ for which the

probability density (or mass) function of each X_i is $f(x_i;\theta)$. Then, the joint probability mass (or density) function of $X_1, X_2, \cdots, X_n$, which we'll (not so arbitrarily) call $L(\theta)$ is:

$$L(\theta) = P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = f(x_1;\theta) \cdot f(x_2;\theta) \cdots f(x_n;\theta) = \prod_{i=1}^{n} f(x_i;\theta)$$

maximum likelihood estimation, one reasonable way to proceed is to treat the "**likelihood function**" $L(\theta)$ as a function of θ, and find the value of θ that maximizes it.

https://online.stat.psu.edu/stat415/lesson/1/1.2

Some Facts

- GDA implies Logistic Regression $p\left(y=1|x\right)=\frac{1}{1+e^{-w^{T}x}}$ but not the other way around.
- If we change Gaussian to Poisson, it also implies Logistic Regression, but not the other way around.
- For a small dataset, it the assumption is “right”, GDA performs better.
- For a large dataset, Logistic will work because it does not assume any distribution (less assumption).

Naïve Bayes Classifier

- Naive Bayes is a classification technique that is based on Bayes' Theorem
- It calculates the probability of each class and then pick the one with the highest probability.

Naïve Bayes Classifier

- As other supervised learning algorithms, naive bayes uses features to make a prediction on a target variable.
- The key difference is that naive bayes assumes that features are independent of each other and there is no correlation between features.
  - However, this is not the case in real life.
- This naive assumption of features being uncorrelated is the reason why this algorithm is called “naive”.

https://towardsdatascience.com/naive-bayes-classifier-explained-50f9723571ed

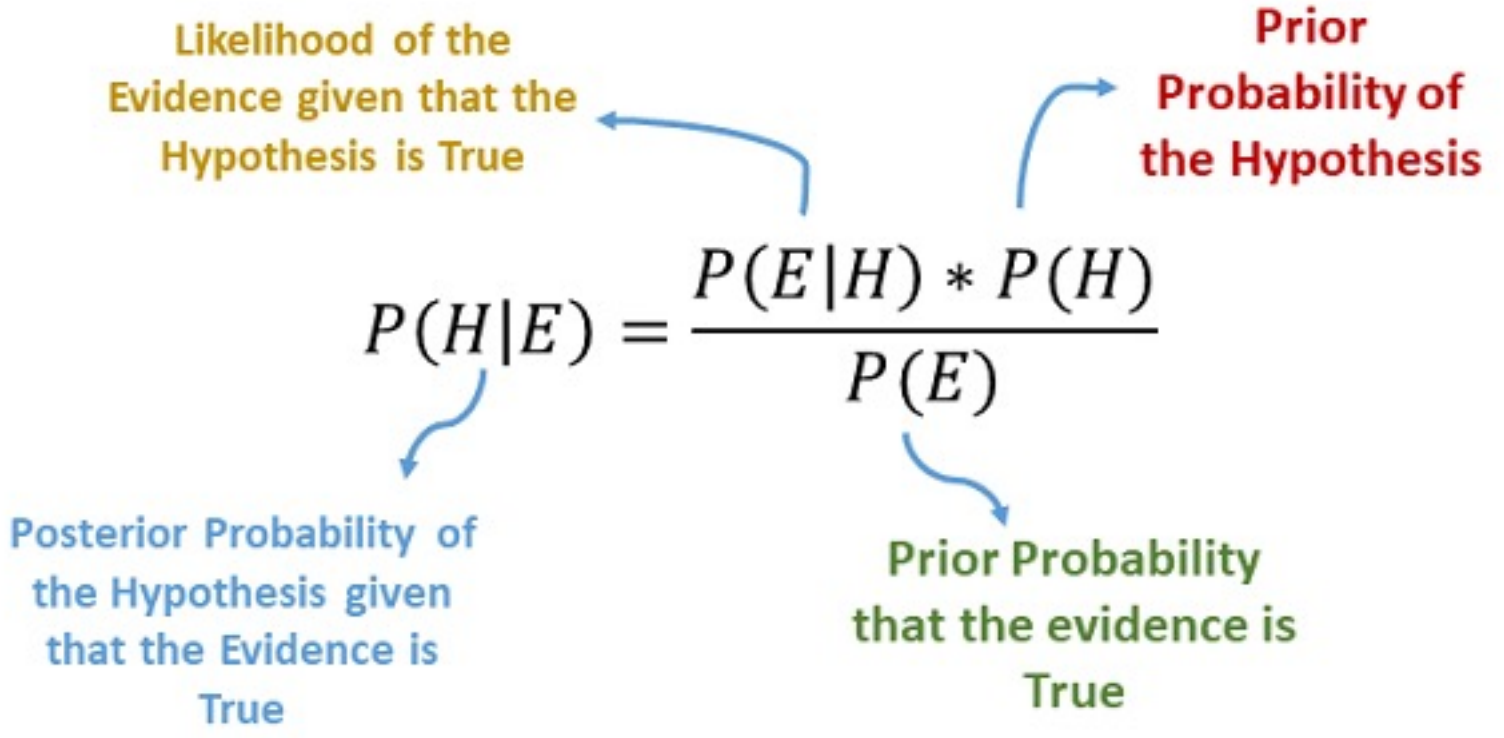
The Naïve Bayes Model

- The *Naïve Bayes Assumption*: Assume that all features are independent **given the class label Y**
- Equationally speaking:

$$P(X_1, \ldots, X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

Bayes Rules

Bayes' Theorem describes the probability of an event, based on a prior knowledge of conditions that might be related to that event.



https://medium.com/analytics-vidhya/na%C3%AFve-bayes-algorithm-5bf31e9032a2

Naïve Bayes

Using the chain rule, the likelihood P(X | y) can be decomposed as:

$$P(X|y) = P(x_1, x_2, \ldots, x_n|y)$$

$$= P(x_1|x_2, \ldots, x_n, y) * P(x_2|x_3, \ldots, x_n, y) \ldots P(x_n|y)$$

but because of the Naive's conditional independence assumption, the conditional probabilities are independent of each other.

$$P(X|y) = P(x_1|y) * P(x_2|y) \ldots P(x_n|y)$$

And as denominator remains constant for all values, the posterior probability can then be:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

https://medium.com/analytics-vidhya/na%C3%AFve-bayes-algorithm-5bf31e9032a2

Naïve Bayes

The Naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that's most probable; this is known as the maximum a posteriori or MAP decision rule.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

https://medium.com/analytics-vidhya/na%C3%AFve-bayes-algorithm-5bf31e9032a2

Why is this useful?

- # of parameters for modeling $P(X_1,\ldots,X_n|Y)$:
  - Need to consider all possibly combinations of X_i, each with 2 parameters (μ, σ).
  - $2(2^n-1)$

- # of parameters for modeling $P(X_1|Y),\ldots,P(X_n|Y)$
  - 2n

Spam/Non-spam Example

- P(Dear|Not Spam) = 8/34
- P(Visit|Not Spam) = 2/34
- P(Dear|Spam) = 3/47
- P(Visit|Spam) = 6/47

	Not Spam	Spam
Dear	8	3
Visit	2	6
Invitation	5	2
Link	2	7
Friend	6	1
Hello	5	4
Discount	0	8
Money	1	7
Click	2	9
Dinner	3	0
Total Words	34	47

Spam/Non-spam Email

assume we have the message "*Hello friend*" and we want to know whether it is a spam or not.

$$P(Hello\ Friend|Not\ Spam) = P(Hello|Not\ Spam) * P(Friend|Not\ Spam)$$

$$P(Not\ Spam|Hello\ Friend) = P(Hello|Not\ Spam) * P(Friend|Not\ Spam) * P(Not\ Spam)$$

$$P(Not\ Spam|Hello\ Friend) = \frac{5}{34} * \frac{6}{34} * \frac{15}{25} = 0.0155$$

$$P(Spam|Hello\ Friend) = \frac{4}{47} * \frac{1}{47} * \frac{10}{25} = 0.00072$$

so, the message "*Hello friend*" is not a spam.

Spam Email

- We are comparing this probability: $p\left(Spam|w_1, w_2, \ldots, w_n\right) = \frac{p(w_1, w_2, \ldots, w_n|Spam)p(Spam)}{p\left(w_1, w_2, \ldots, w_n\right)}$ to $p(nonSpam|w_1, w_2, \ldots, w_n)$.
- Assume $w_i^{'}s$ are i.i.d. and ignore the constant, we are computing $\mathrm{p(Spam)}\Pi_{i=1}^{n} p(w_i|Spam)$ and $\mathrm{p(nonSpam)}\Pi_{i=1}^{n} p(w_i|nonSpam)$
- $p(Spam)$ is the ratio of # of spam email to total # of messages.
- $p(nonSpam) = 1 - p(Spam)$

Spam Email (cont.)

- $p(w_i|Spam)$ is the probability of the word w_i occurs in spam messages.
- $p(w_i|Spam) = \frac{\#\ of\ w_i\ in\ spam\ messages}{total\ \#\ of\ words\ in\ spam\ messages}$
- However, some word may not appear in the training dataset. So it may nullify the probability.
- $p(w_i|Spam) = \frac{\#\ of\ w_i\ in\ spam\ messages+\alpha}{total\ \#\ of\ words\ in\ spam\ messages+\alpha N}$ where N is the number of unique words in the whole dataset.

Handwriting Digit Classification

- The dataset in MNIST contains 10 digit images.
- We may use naïve Bayes classifier for two digits.

Assume All Input Dimensions of x is Independent

- $x_i \sim G\left(\mu_i, \sigma_i^2\right) \forall\ i = 784$
- $P(x|c) = \prod_{\{i=1\}}^{784} \frac{1}{\sqrt{2\ \pi\ \sigma_i^2}} \exp(-\frac{\frac{1}{2}(x_i - \mu_i)^2}{\sigma_i^2})$
- We need to estimate $\mu_i\ and\ \sigma_i$ with respect to a class c.

Cont.

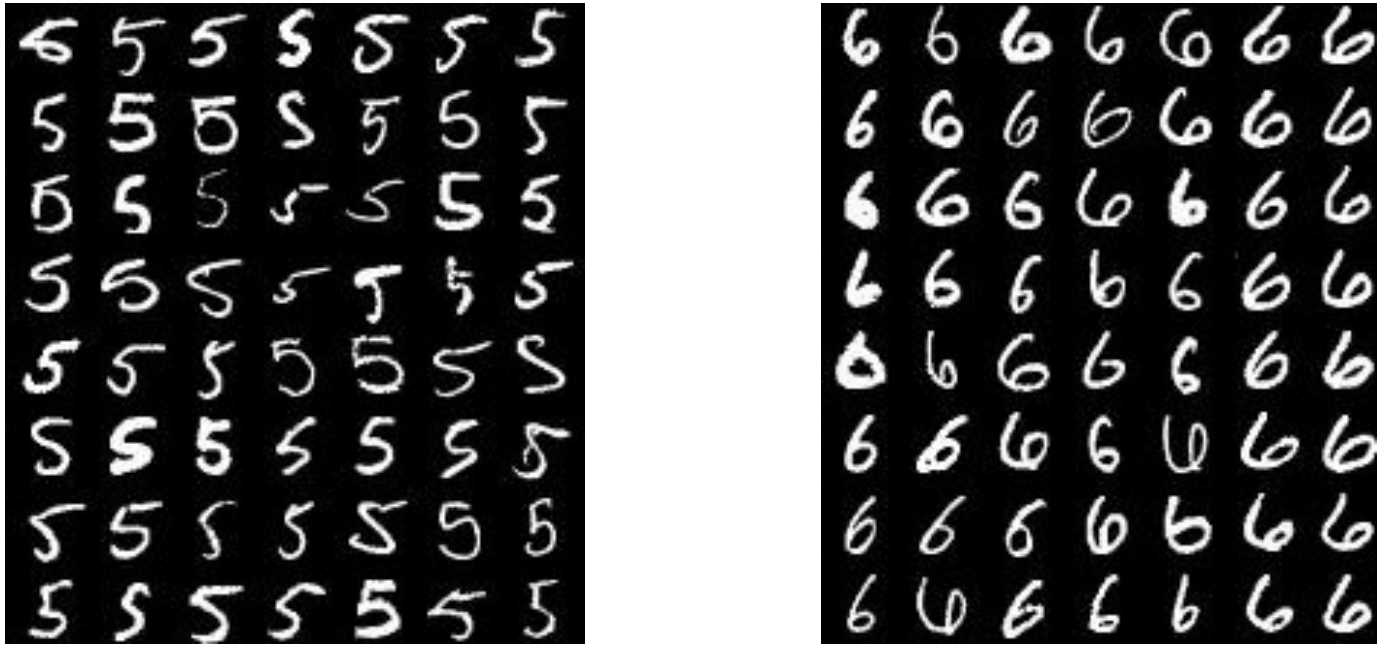
- If we want to calculate the mean and covariance for all the images of the digit 9, then we'd first grab only the images of the digit 9 (ignoring the rest of the images), and calculate the sample mean and covariance from this subset.
- Because probabilities are very small when dimensionality is high, we'll work with log-likelihood instead of likelihood.
- So given x, we are computing the probability $p(c|x) = \frac{p(x|c)p(c)}{p(x)}$ with respect to each class c. $p(x)$ is a constant. We only need to compute $p(x|c)p(c)$.
- Using the log probability for prediction:

$$c^* = \arg max_c \log(P(x|c)) + \log P(c))$$

- $\log(p(x|c) = \sigma_{\{i=1\}}^{784} \log(\frac{1}{\sqrt{2\,\pi\,\sigma_i^2}} \exp\left(-\frac{\frac{1}{2}(x_i-\mu_i)^2}{\sigma_i^2}\right))$

Naïve Bayes Training

- Now that we've decided to use a Naïve Bayes classifier, we need to train it with some data:



MNIST Training Data

Naïve Bayes Training

- Training in Naïve Bayes is **easy**:
  - Estimate P(Y=v) as the fraction of records with Y=v

$$P(Y = v) = \frac{Count(Y = v)}{\#\ records}$$

  - Estimate $P(X_i=u|Y=v)$ as the fraction of records with Y=v for which $X_i=u$

$$P(X_i = u|Y = v) = \frac{Count(X_i = u \wedge Y = v)}{Count(Y = v)}$$

- (This corresponds to Maximum Likelihood estimation of model parameters)

Naïve Bayes Training

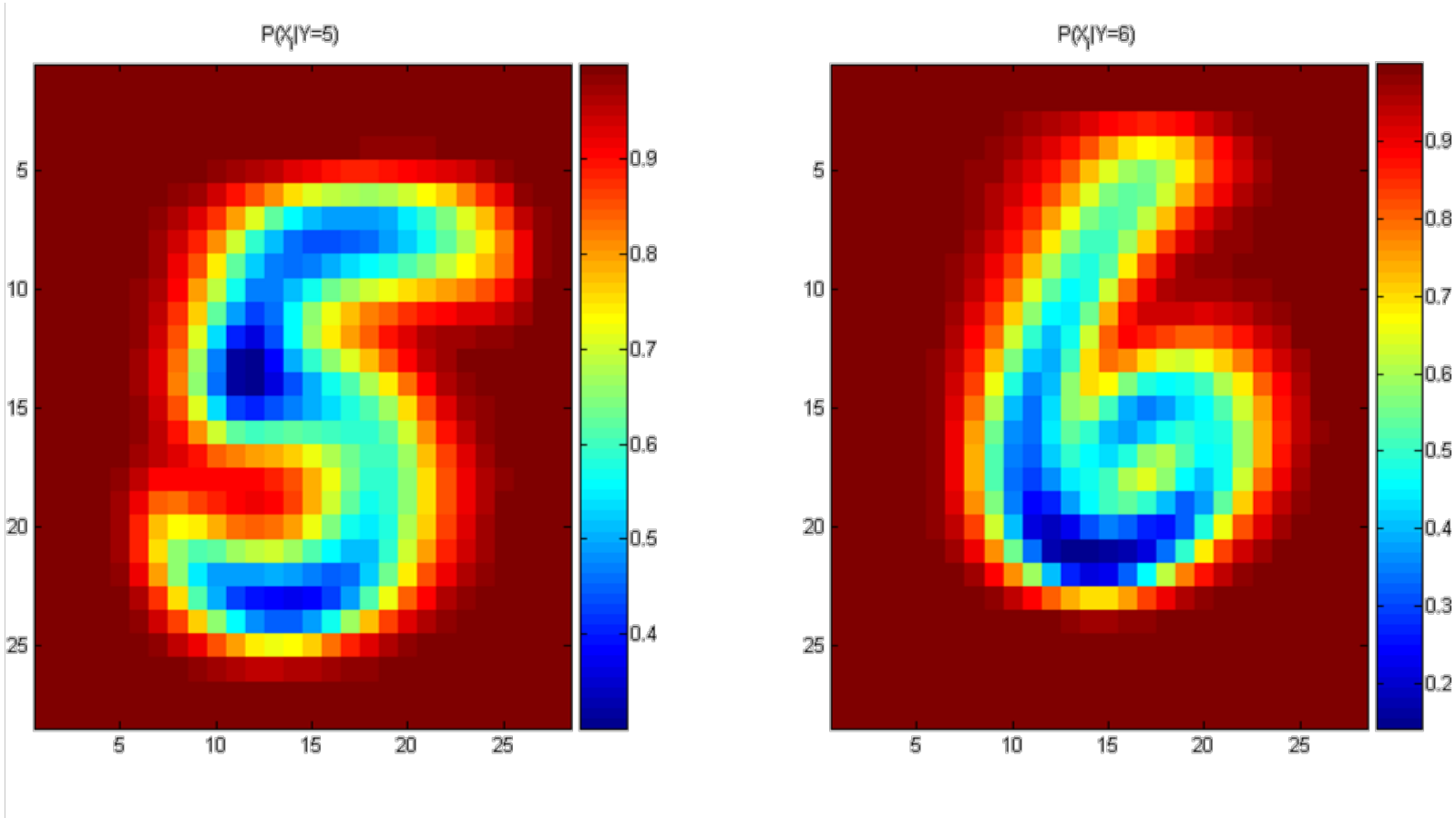
- In practice, some of these counts can be zero
- Fix this by adding "virtual" counts:

$$P(X_i = u|Y = v) = \frac{Count(X_i = u \wedge Y = v) + 1}{Count(Y = v) + 2}$$

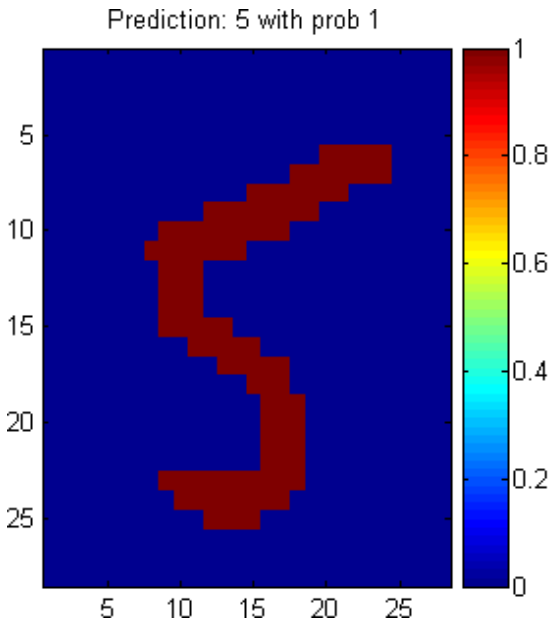
- (This is like putting a prior on parameters and doing MAP estimation instead of MLE)
- This is called *Smoothing*

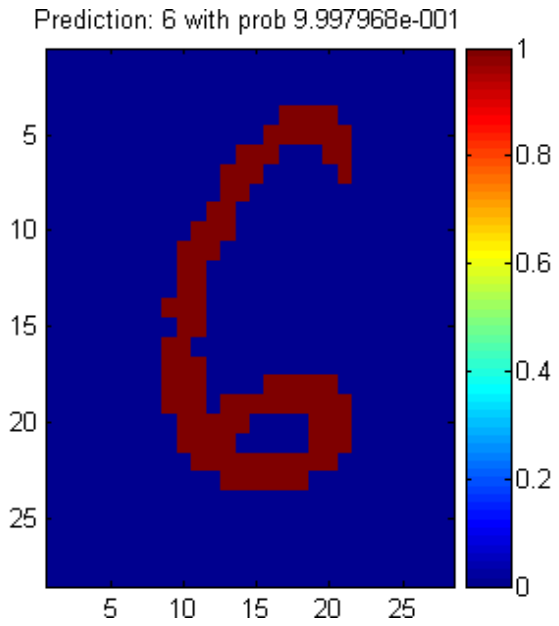
Naïve Bayes Training

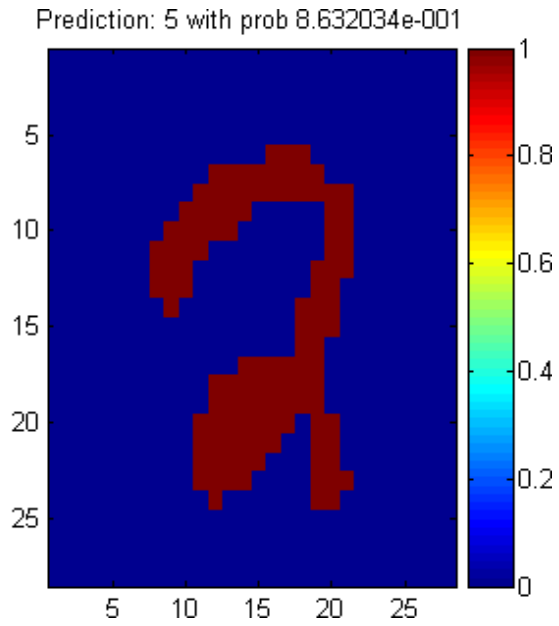
- For binary digits, training amounts to averaging all of the training fives together and all of the training sixes together.



Naïve Bayes Classification







Another Example of the Naïve Bayes Classifier

The weather data, with counts and probabilities

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

A new day

outlook	temperature	humidity	windy	play
sunny	cool	high	true	?

- Likelihood of yes

$$= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$

- Likelihood of no

$$= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

- Therefore, the prediction is No

The Naive Bayes Classifier for Data Sets with Numerical Attribute Values

- One common practice to handle numerical attribute values is to assume normal distributions for numerical attributes.

The numeric weather data with summary statistics

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3		83	85		86	85	false	6	2	9	5
overcast	4	0		70	80		96	90	true	3	3		
rainy	3	2		68	65		80	70					
				64	72		65	95					
				69	71		70	91					
				75			80						
				75			70						
				72			90						
				81			75						
sunny	2/9	3/5	mean	73	74.6	mean	79.1	86.2	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	std dev	6.2	7.9	std dev	10.2	9.7	true	3/9	3/5		
rainy	3/9	2/5											

- Let x_1, x_2, …, x_n be the values of a numerical attribute in the training data set.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$$

$$f(w) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(w-\mu)^2}{\sigma^2}}$$

- For examples,

$$f(\text{temperature} = 66 \mid \text{Yes}) = \frac{1}{\sqrt{2\pi}(6.2)} e^{-\frac{(66-73)^2}{2(6.2)^2}} = 0.0340$$

- Likelihood of Yes = $\frac{2}{9} \times 0.0340 \times 0.0221 \times \frac{3}{9} \times \frac{9}{14} = 0.000036$

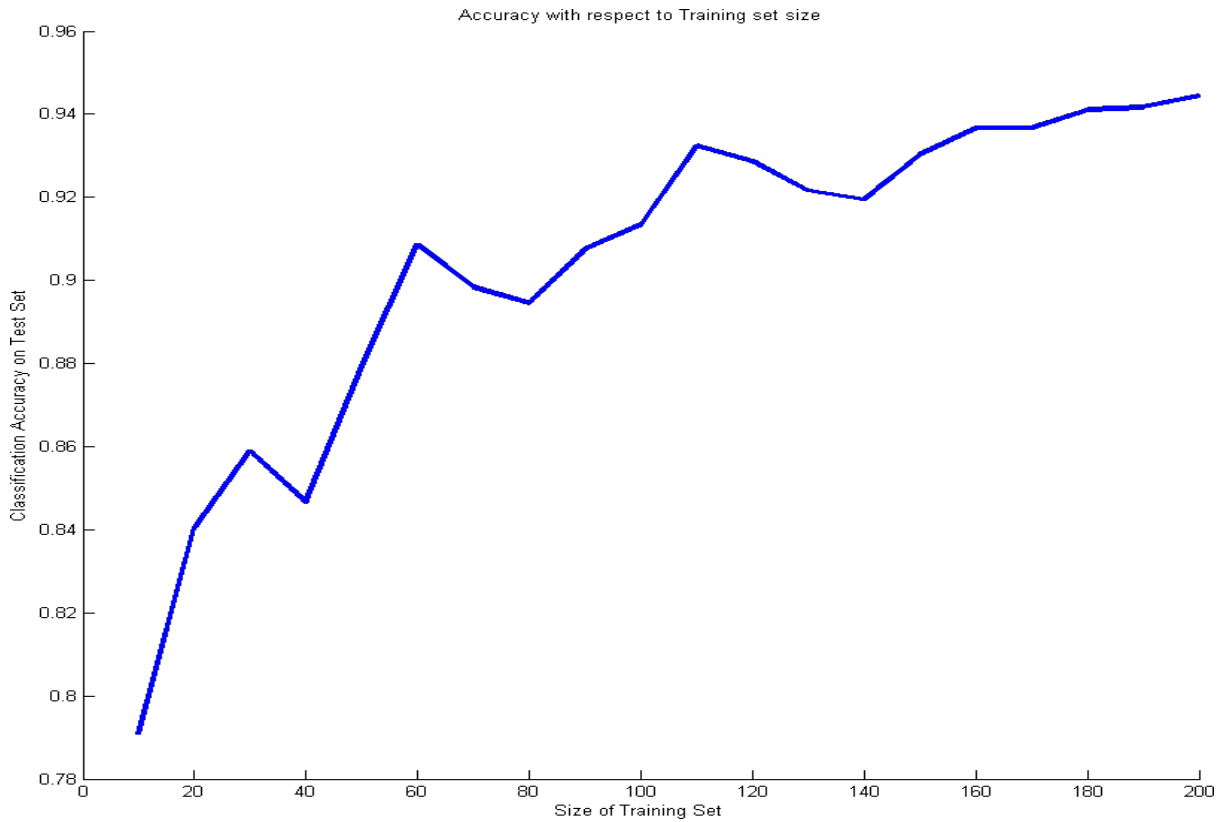
- Likelihood of No = $\frac{3}{5} \times 0.0291 \times 0.038 \times \frac{3}{5} \times \frac{5}{14} = 0.000136$

Outputting Probabilities

- What's nice about Naïve Bayes (and generative models in general) is that it returns probabilities
  - These probabilities can tell us how confident the algorithm is
  - So… don't throw away those probabilities!

Performance on a Test Set

- Naïve Bayes is often a good choice if you don't have much training data!



Naïve Bayes Assumption

- Recall the Naïve Bayes assumption:
  - that all features are independent **given the class label Y**

- Does this hold for the digit recognition problem?

Exclusive-OR Example

- For an example where conditional independence fails:
  - $Y=XOR(X_1,X_2)$

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

Naïve Bayes Assumption

- Actually, the Naïve Bayes assumption is almost never true

- Still… Naïve Bayes often performs surprisingly well even when its assumptions do not hold

Numerical Stability

- It is often the case that machine learning algorithms need to work with very small numbers
  - Imagine computing the probability of 2000 independent coin flips
  - MATLAB thinks that $(.5)^{2000}=0$

Underflow Prevention

- Multiplying lots of probabilities

➔ floating-point underflow.

- Recall: *log(xy) = log(x) + log(y),*

➔ better to sum logs of probabilities rather than multiplying probabilities.

Underflow Prevention

- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

Numerical Stability

- Instead of comparing P(Y=5|X_1,…,X_n) with P(Y=6|X_1,…,X_n),
  - Compare their logarithms

$$\begin{aligned}
\log\left(P(Y|X_1,\ldots,X_n)\right) &= \log\left(\frac{P(X_1,\ldots,X_n|Y)\cdot P(Y)}{P(X_1,\ldots,X_n)}\right) \\
&= \text{constant} + \log\left(\prod_{i=1}^{n} P(X_i|Y)\right) + \log P(Y) \\
&= \text{constant} + \sum_{i=1}^{n} \log P(X_i|Y) + \log P(Y)
\end{aligned}$$

Recovering the Probabilities

- What if we want the probabilities though??
- Suppose that for some constant K, we have:

$$\log P(Y = 5|X_1, \ldots, X_n) + K$$

- And

$$\log P(Y = 6|X_1, \ldots, X_n) + K$$

- How would we recover the original probabilities?

Recovering the Probabilities

- Given: $\alpha_i = \log a_i + K$
- Then for any constant C:

$$\begin{aligned} \frac{a_i}{\sum_i a_i} &= \frac{e^{\alpha_i}}{\sum_i e^{\alpha_i}} \\ &= \frac{e^C \cdot e^{\alpha_i}}{\sum_i e^C \cdot e^{\alpha_i}} \\ &= \frac{e^{\alpha_i + C}}{\sum_i e^{\alpha_i + C}} \end{aligned}$$

- One suggestion: set C such that the greatest α_i is shifted to zero:

$$C = -\max_i \{\alpha_i\}$$

See https://stats.stackexchange.com/questions/105602/example-of-how-the-log-sum-exp-trick-works-in-naive-bayes?noredirect=1&lq=1

Pros and Cons of Naive Bayes Algorithm

Pros:

- The assumption that all features are independent makes naive bayes algorithm **very fast** compared to complicated algorithms. In some cases, speed is preferred over higher accuracy.
- It works well with high-dimensional data such as text classification, email spam detection.

Cons:

- The assumption that all features are independent is not usually the case in real life so it makes naive bayes algorithm less accurate than complicated algorithms. Speed comes at a cost!

https://towardsdatascience.com/naive-bayes-classifier-explained-50f9723571ed

Recap

- We defined a *Bayes classifier* but saw that it's intractable to compute $P(X_1,\ldots,X_n|Y)$
- We then used the *Naïve Bayes assumption* – that everything is independent given the class label Y
- A natural question: is there some happy compromise where we only assume that *some* features are conditionally independent?
  - Perhaps

Conclusions

- Naïve Bayes is:
  - Really easy to implement and often works well
  - Often a good first thing to try
  - Commonly used as a “punching bag” for smarter algorithms