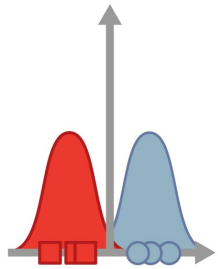


# Estimating Probability Distributions

## What you'll do

- Approximate distributions from data with Maximum Likelihood Estimate (MLE)
- Use Naive Bayes assumption to estimate probabilities from high dimensional data
- Implement the Naive Bayes algorithm to build a baby name classifier



## Course Description

Knowing the probability distribution of data allows you to make highly accurate predictions about future data points. Unfortunately, it is not often that you know the probability distribution of your data. You can, however, use techniques that allow you to estimate the probability

distribution.

In this course, Professor Weinberger introduces you to Maximum Likelihood Estimate (MLE), which you will use to approximate distributions from data. You will also investigate the Bayes optimal classifier and identify how the assumptions you make about your data will impact these estimations. You will then apply the Naive Bayes assumption to estimate probabilities for problems that contain a high number of dimensions. Ultimately, you will apply these principles and techniques to implement a Naive Bayes classifier that can determine the common gender of baby names based on a set of features that capture common characteristics of names.

**System requirements:** This course contains a virtual programming environment that does not support the use of Safari, IE, Edge, tablets, or mobile devices. Please use Chrome or Firefox on a computer for this course.



**Kilian Weinberger**  
**Associate Professor**  
**Computing and Information Science, Cornell University**

**Kilian Weinberger** is an Associate Professor in the Department of Computer Science at Cornell University. He received his Ph.D. from the University of Pennsylvania in Machine Learning under the supervision of Lawrence Saul and his undergraduate degree in Mathematics and Computer Science from the University of Oxford.

During his career, he has won several best paper awards at ICML (2004), CVPR (2004, 2017), AISTATS (2005) and KDD (2014, runner-up award). In 2011, he was awarded the Outstanding AAAI Senior Program Chair Award and in 2012 he received an NSF CAREER award. He was elected co-Program Chair for ICML 2016 and for AAAI 2018. In 2016, he was the recipient of the Daniel M Lazar '29 Excellence in Teaching Award.

Professor Weinberger's research focuses on Machine Learning and its applications. In particular, he focuses on learning under resource constraints, metric learning, machine-learned web-search ranking, computer vision, and deep learning. Before joining Cornell University, he was an Associate Professor at Washington University in St. Louis and previously worked as a research scientist at Yahoo! Research.

## Table of Contents

---

### Q and A

[Live Labs: Building a Community of Learners](#)  
[Live Labs Q&A](#)

### Module 1: Estimating Distributions from Data

[Module Introduction: Estimating Distributions from Data](#)

Read: Statistics Refresher

Watch: Data distributions and the Bayes Optimal Classifier

Read: Bayes Optimal Classifier

Tool: Bayes Optimal Classifier Cheat Sheet

Watch: Estimate Distribution with Maximum Likelihood Estimation

Read: Derivation of MLE for Binomial Distribution

Ask the Expert: Minmin Chen

Watch: Smoothing for Small Data Sets

Activity: Test the Effect of Smoothing

Module Wrap-up: Estimating Distributions from Data

## Module 2: Classification with the Naive Bayes Algorithm

Module Introduction: Classification with the Naive Bayes Algorithm

Watch: Limitations of High Dimensional Space in MLE

Read: MLE Curse of Dimensionality

Read: Formalize the Curse of Dimensionality

Activity: Capturing All Possibilities in  $d$  Dimensions

Watch: Naive Bayes Assumption

Data Sets Where the Naive Bayes Assumption Holds

Watch: Naive Bayes Classifier

Read: Derivation of Naive Bayes Classifier

Tool: Naive Bayes Cheat Sheet

Watch: Determine Probability With Categorical Naive Bayes

Read: Categorical Naive Bayes Classifiers

Activity: Naive Bayes in Action

Module Wrap-up: Classification with the Naive Bayes Algorithm

## Module 3: Building a Baby Name Classifying System

Module Introduction: Building a Baby Name Classifying System

Watch: Build a Baby Name Classifier

Watch: Determine Class Prior Probability

Code: Class Prior Probability

Watch: Determine Conditional Probability

Code: Conditional Probability

Watch: Create Feature Vectors from Data

Code: Feature Extraction and Hashing

Build a Baby Name Classification System

Improving Focus on Features

Module Wrap-up: Building a Baby Name Classifying System

Read: Thank You and Farewell

# Q&A

1. [Live Labs: Building a Community of Learners](#)
2. [Live Labs Q&A](#)

---

[Back to Table of Contents](#)

# Live Labs: Building a Community of Learners

## Building a Community of Learners

This course will include two live (synchronous) video sessions called Live Labs. Live Labs will be offered each week as an opportunity to connect with your peers and instructor to address questions about the coursework and dive deeper into the material. Each hour-long session will be scheduled in advance. You are highly encouraged to take advantage of this chance to connect with your community of learners to build relationships and make this course experience even more rewarding.

This short Q&A section of the course contains a discussion called **Live Labs Q&A** where you can post questions and interact with your peers at any time as you work through this course. Common questions and important topics brought up in this discussion may be addressed in the Live Labs sessions.

In order to review the session schedule and join a session, you'll need to navigate to the **Live Labs** link under the **Course Shortcuts** section in the navigation menu on the left,

---

[Back to Table of Contents](#)

## Live Labs Q&A

The Live Labs Q&A discussion is a place for you to post questions and interact with your peers throughout your work in this course. Common questions and important topics brought up in this discussion will likely be addressed in a Live Labs session.

### Instructions:

Read through existing posts to find out what other students are discussing. Upvote posts that you find interesting or contain questions that you also have by "liking" it with the thumbs-up button at the bottom of the post.

If you have something to contribute to an existing discussion, reply to the thread and get involved in the conversation.

If you have new ideas, questions, or issues, post them in this Live Labs discussion board. If you have potential answers or suggestions that may resolve a question thread, feel free to share your understanding in a reply. This will help guide the instructor's feedback and discussion during their live session.

**The questions that are the most upvoted or replied to that haven't been resolved will likely be addressed at the scheduled time by the instructor.**

---

---

[Back to Table of Contents](#)

# Module 1: Estimating Distributions from Data

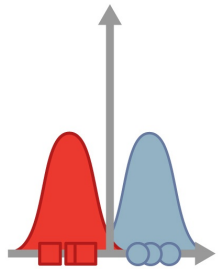
1. [Module 1 Introduction: Estimating Distributions from Data](#)
2. [Read: Statistics Refresher](#)
3. [Watch: Data distributions and the Bayes Optimal Classifier](#)
4. [Read: Bayes Optimal Classifier](#)
5. [Tool: Bayes Optimal Classifier Cheat Sheet](#)
6. [Watch: Estimate Distribution with Maximum Likelihood Estimation](#)
7. [Read: Derivation of MLE for Binomial Distribution](#)
8. [Ask the Expert: Minmin Chen](#)
9. [Watch: Smoothing for Small Data Sets](#)
10. [Activity: Test the Effect of Smoothing](#)
11. [Module 1 Wrap-up: Estimating Distributions from Data](#)

---

[Back to Table of Contents](#)



# Module 1 Introduction: Estimating Distributions from Data



In this module, you will investigate the Bayes optimal classifier, which is the optimal way to predict the labels of data points if the data's distribution is known. As typically the data distribution is not known, you will explore a method of estimating data distributions empirically. You will estimate the probability distribution of a coin flip from training samples of coin flips, using a method known as

Maximum Likelihood Estimation (MLE). These concepts build a foundation for overcoming the challenges of estimating data distributions in high dimensions that will be covered in later modules.

---

[Back to Table of Contents](#)

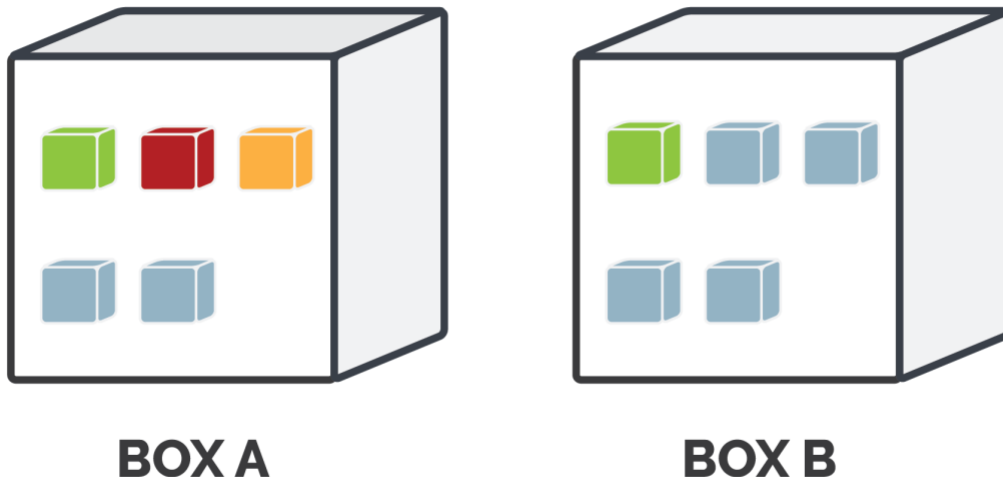
## Read: Statistics Refresher

### Bayes' Rule

Bayes' rule is used to describe the probability of an event based on the probability of other conditions. Bayes' theorem is stated mathematically as the following equation:

$$P(A/B) = P(B/A)P(A)/P(B)$$

where A and B are events and  $P(B)$  is not equal to 0.  $P(A/B)$  is the likelihood of event A given B.  $P(A)$  is the likelihood of observing the event independently.



Imagine you have two boxes, Box A and Box B, each containing a set of different colored blocks.

Box A contains one green block, one red block, one orange block, and two blue blocks.

Box B contains one green block, and four blue blocks.

Imagine you are blindfolded and you randomly select a box and pull a single block from that box. If the block is blue, what is the probability that you had picked from Box A?

To solve this, you can use Bayes' rule:

$$P(\text{Box A}|\text{blue})=P(\text{blue}|\text{Box A})P(\text{Box A})/P(\text{blue})$$

If you replace the terms with their actual probabilities and solve, you should get the following:  $P(\text{Box A}|\text{blue})=(2/5)(1/2)/6/10=1/3$

### Chain Rule

The chain rule allows you to calculate the probability of multiple events using conditional probabilities. Consider four events: A, B, C, and D. We can express the joint probability of the four events as such:

$$P(A,B,C,D)=P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$



Using the previous block example, imagine you place all of the blocks in

one box. Your box now contains two green blocks, one red block, one orange block, and six blue blocks.

Let's define the following events:

A = pull blue block

B = pull red block

C = pull orange block

D = pull green block

If you want to find the probability that you will pull blocks in this sequence, without replacing the block after pulling it, you should arrive at the following:

$$P(A,B,C,D)=(6/10)(1/9)(1/8)(2/7)=(1/420)$$

## Distributions

### Normal distribution

A random variable  $x$  that follows a normal distribution has the following probability density function:

$$f(x;\mu,\sigma)=1/\sqrt{2\pi\sigma^2}e^{-(x-\mu)^2/2\sigma^2}$$

- The expectation of a normal distributed random variable is  $\mu$  and the variance is  $\sigma^2$ .
- This is commonly seen in nature and models the “bell curve.”; heights, for example, are normally distributed.
- Note that the probability density function of the normal distribution has a maximum at the mean.
- The normal distribution is sometimes referred to as the Gaussian distribution.

### Binomial distribution

- A random variable  $x$  that follows a binomial distribution has the following probability mass function:

$$P(x=k;p) = \binom{n}{k} p^k (1-p)^{n-k}$$

- This discrete distribution describes the probabilities of obtaining  $k$  successes from  $n$  independent trials with two possible outcomes, e.g. the number of tails from  $n$  coin flips.
- The expectation of  $x$  is  $np$ .
- The variance of  $x$  is  $np(1-p)$ .
- It is used to model the outcome of  $n$  coin flips.

## Watch: Data distributions and the Bayes Optimal Classifier

If you know or can estimate your data distribution, you can make accurate predictions about the label of a new data point. In this video, Professor Weinberger describes the Bayes optimal classifier, which is used to assign labels to data with a known probability distribution. For a given input, the Bayes optimal classifier will select a label that is most likely. Typically, the data distribution is not known, but often it can be estimated.

### Video Transcript

The data that we work with is always drawn from some probability distribution. This distribution is typically not known to us, but we do have that data. If we knew that distribution, we wouldn't actually need the data, right? If you knew the distribution, you could make decisions based on the distribution. Let me give you an example. So,  $P$  of  $x$  is probability over a certain input; let's say we're talking about spam classification, all right?  $x$  is an email, so  $P$  of  $x$  is the probability that an email arrives in my inbox. Probability  $P$  of  $y$  given  $x$  is the probability that I consider it spam, given that that email arrived in my inbox. For certain email, that may not be deterministic, because some people may consider email to be spam, others may consider it not spam. That's why spam works; it's because some people consider it valuable information. So let's say we have a probability of  $P$  of  $y$  given  $x$  that the email is spam to be 90%, and 10% of the people may consider it not to be spam. What decision would we make? Well, given that it's so overwhelmingly spam, the best decision would be to say this is a spam email because you will be right 90% of the time, and you would only be wrong 10% of the time. That's called the Bayes optimal classifier. The Bayes optimal classifier looks at the distribution  $P$  of  $y$  given  $x$  and then assigns the most common label to that instance. Unfortunately, we don't have the distribution and we can't use the Bayes optimal classifier. But what we can do is we can take the data that we have observed and use that data to estimate the distribution  $P$  of  $y$  given  $x$ , and then use our estimate to behave as if it was the real distribution.

## Check Your Knowledge

---

[Click to check your knowledge of data distribution and the Bayes optimal classifier.](#)

[Back to Table of Contents](#)

## Read: Bayes Optimal Classifier

### Formalizing the Bayes Optimal Classifier

The Bayes optimal classifier simply assigns the most likely label for a given(test) input. More formally, it is based on the conditional distribution  $P(y|x)$ , predicting the most likely label for a new data point ( $x$ ). You make a prediction ( $y^*$ ) by finding the arguments of the maxima ( $\arg \max$ ) of  $P(y|x)$ . In this way, the predicted value optimizes a function  $h(x)$ .

The Bayes optimal classifier predicts:

$$y^* = h_{\text{opt}}(x) = \arg \max_y P(y|x)$$

### Computing the Error

This classifier cannot be used in practice, since you will rarely know the true underlying data distribution. However, it provides a highly informative lower bound of the error rate when you estimate the distribution of your data. With the same feature representation, no classifier can obtain a lower error. As such, it is sometimes referred to as the "irreducible error."

Although the Bayes optimal classifier is as good as it gets, it will make an incorrect prediction if a sample does not have the most likely label. You can compute the probability of that happening precisely (which is exactly the error rate):

$$\epsilon_{\text{BayesOpt}} = 1 - P(h_{\text{opt}}(x)|x) = 1 - P(y^*|x)$$

Essentially, the Bayes Optimal classifier is wrong whenever the most likely outcome conditional on a set of features,  $y^*$ , does not happen. This event has the probability  $1 - P(y^*|x)$ .

You want to classify an email as either spam (+1) or not spam (-1). Let's assume that this email ( $x$ ) has the following conditional class probabilities:

$$P(+1|x) = 0.8 \text{ (e.g. 80\% probability it is spam)}$$

$$P(-1|x) = 0.2 \text{ (e.g. 20\% probability it is not spam)}$$



[Click to check your knowledge of the Bayes optimal classifier.](#)

## Tool: Bayes Optimal Classifier Cheat Sheet

Use this [Bayes Optimal Classifier Cheat Sheet](#) as a quick way to review the details of this classifier.

This tool provides an overview of the Bayes optimal classifier for your reference. You'll find information about the applicability, underlying mathematical principles, assumptions, and other details of this classifier.

---

[Back to Table of Contents](#)

## Watch: Estimate Distribution with Maximum Likelihood Estimation

Since it's unlikely that you'll know your data's distribution, you'll instead need a way to estimate the distribution. Maximum Likelihood Estimation, or MLE, is a way to estimate parameters of a distribution from data samples. In this video, Professor Weinberger uses flipping a coin and the outcome of heads or tails as an example of how we might apply MLE to empirically estimate the bias of a coin.

### Video Transcript

Maximum Likelihood Estimation is one way to estimate a distribution from data that's drawn from that distribution. So, machine learning, we have exactly that setting; we have some distribution that we don't know, but the training data was drawn from that distribution. One thing we can do now is we can take that data, estimate the distribution, and then use the Bayes optimal classifier on that estimate. So, Maximum Likelihood Estimation works as follows: We assume we know something about that distribution; for example, its form, what kind of family it belongs to. And these families typically have some parameters. And then we tweak these parameters such that the data that we actually observed becomes as likely as possible. And the assumption basically is, "Well, if I observed it, it must be pretty likely."

Let me go through an example. So, let's say we have some data,  $x$  and  $y$ , and we would like to estimate  $P$  of  $y$  given  $x$ , because that's ultimately what we need for decision-making. For example,  $x$  could be an email, and  $y$  could be, is it spam or not spam? Well, if you only have two binary decisions, like spam or not spam — but actually, these problems are exactly equivalent to a coin toss, right? So  $x$  describes my coin, and when I draw a label for this particular instance, I'm basically tossing the coin, and either it comes up heads — that means it's positive — or tails means it's negative. What we're trying to estimate is the weighing of the coin; how likely is it that it's heads? So let's call this  $\theta$ , So  $\theta$  is our parameter and it's the probability that this particular coin comes up heads, and the probability that it comes up tails is  $1 - \theta$ . How would

we estimate this from  $\theta$ ? Well, we just look at how many times was that coin tossed? So, for example, this case, assume we observed ten tosses of this particular coin, and out of these ten, six times it came up heads, and four times it came up tails. Just to connect it to the email example, imagine we had the same email got sent to ten people; six thought it was spam and four thought it was not spam. Now we would like to estimate the parameter  $\theta$  from this data. And what is the probability of the data that we observed? Well, actually, if we know the probability of heads is  $\theta$ , then the probability of the data is just  $\theta$  to the power of  $h$ . Because if  $\theta$  is the probability that it comes up heads to the power of how many times we saw heads, times the probability  $1 - \theta$ , which is the probability of tails, to the power of how many times we saw tails, that term is what we want to maximize. That's the probability that we observed the data that we saw. To maximize this, while this has a lot of products in it and powers — and those are kind of hard to deal with, so the first thing you do is you just take a log. Because if you take a log of a function, the maximum doesn't change. So that's an old trick to make the math easier. So we take the log; that turns into  $h \times \log \theta$  plus  $t \times \log (1 - \theta)$ . And now we have a function we want to maximize that's much better behaved, and in order to maximize a function, what do we do? We take the first derivative and set it to zero; that should give us the maximum of that function. So if you take the derivative and solve that and set it to zero, we can solve that expression with respect to  $\theta$ , and we get  $\theta$  equals  $h$  divided by  $t$  plus  $h$ . What is  $h$ ? The number of times we toss heads. What's  $t$  plus  $h$ ? The total number of times we tossed the coin. So we get 0.6, so 60% chance it comes up heads. And that's a very intuitive answer. And so if you have a coin that's tossed ten times and then comes six times heads, you would say 60% is probably a good estimate for the probability of heads. So, formally what we're trying to do is we assume there's some distribution — some parametric form of the distribution  $P$  of  $\theta$ , of  $y$  given  $x$ , and then we say we tweak  $\theta$  such that we maximize the probability of  $y_i$  given  $x_i$  for every single  $i$  in our data set. So we take the big product and maximize this. And that's always the same two steps we go through; we take the log and then we take the derivative, equate it to zero, and solve it.

[Click to check your knowledge of distribution with MLE.](#)

[Back to Table of Contents](#)

## Read: Derivation of MLE for Binomial Distribution

You can use MLE to estimate the parameters of a probability distribution, given the data.

To maximize the probability, take the derivative of the likelihood function and equate it with 0.

Calculating derivatives is cumbersome when a function is a product of many terms. However, you can apply the log function and maximize the log-likelihood instead.

### Formalizing Maximum Likelihood Estimation

Let's formally verify the result obtained in the previous coin toss example.

Namely, given multiple coin tosses resulting in  $n_T$  tails and  $n_H$  heads, the MLE estimate of the probability that a coin comes up heads is:

$$P(H) = n_H / (n_H + n_T)$$

Let us denote the probability of obtaining heads as  $P(H) = \theta$ . Further, assume you have observed the data  $D$  (sequence of heads and tails). You would like to infer the true value of  $\theta$  from the observed data  $D$ .

For any given probability  $\theta$ , you can compute the likelihood of the observed data  $P(D|\theta)$ . The MLE principle estimates the probability of heads  $\theta$  so that it maximizes the likelihood of the observed data. The resulting estimate is referred to as the maximum likelihood estimate of  $\theta$ :

$$\theta = \arg \max_{\theta} P(D|\theta)$$

### Coin Toss Example

Coming back to our coin toss example, you are interested in estimating  $P(H)=\theta$  using the maximum likelihood principle. You know that the result of coin tosses follows the binomial distribution, so you can model the following for some parameter  $\theta$ :

$$P(D|\theta) = \binom{n}{H} \theta^H (1-\theta)^{n-H}$$

You are interested in finding the maximizer of  $P(D|\theta)$ . In order to do so, you are going to take the derivative of the likelihood function and equate it with 0. The function is a product of many terms, which makes the calculation of derivatives cumbersome. However, you can deploy a trick to find the maximizer and apply the  $\log()$  function to maximize the log-likelihood instead:

$$\arg \max_{\theta} P(D|\theta) = \arg \max_{\theta} \log P(D|\theta)$$

Note that the log function is monotonically increasing and therefore the maximizer of  $P(D|\theta)$  is also the maximizer for  $\log P(D|\theta)$ . The  $\log()$  also turns any product into sums, so you obtain:

$$\log P(D|\theta) = \log \binom{n}{H} + H \log \theta + (n-H) \log (1-\theta)$$

Now we'll compute the derivative of the log likelihood with respect to  $\theta$  and equate it with zero (note that the first term is constant and drops out):

$$0 = H \frac{\partial \log \theta}{\partial \theta} - (n-H) \frac{\partial \log (1-\theta)}{\partial \theta} \Rightarrow \frac{H}{\theta} = \frac{n-H}{1-\theta} \Rightarrow H(1-\theta) = (n-H)\theta \Rightarrow \theta = \frac{H}{n}$$

This final form aligns with the intuition that:

$$P(H) = \frac{H}{n}$$

## Maximum A Posteriori Estimation

Maximum a posteriori estimation (MAP) is a generalization of the maximum likelihood estimator that incorporates a prior belief regarding the distribution of the parameter of interest, in this case  $P(\theta)$ .

$$\theta = \arg \max_{\theta} P(D|\theta)P(\theta)$$

You should recognize the expression on the right as the unnormalized posterior from Bayes Rule. You can see that if  $P(\theta)$  is uniform over  $\theta$  — the same thing as having no prior information — the expression reduces to the MLE expression derived above.



## Ask the Expert: Minmin Chen

---



**Minmin Chen** is a Research Scientist at Google. Her main research interests are in machine learning, with a focus on sequence modeling and reinforcement learning for recommendation systems. Previously, she was a research scientist at Criteo Lab, building computational models for online advertising, as well as Amazon, working on the Amazon Go project. She completed her PhD studies at Washington University in St. Louis on representation learning and domain adaptation. She has published over twenty papers at top conferences in machine learning such as NIPS, ICML, ICLR and AISTATs.

---

### How are MLE and MAP used at Google?

So, maximum likelihood estimation and MAP are very heavily used within Google. So, I mean, in our case, for example, for video recommendations, you would need to — I mean, one of the main predictions you need to make is, for example, how likely a user will click or watch a video, and that's done using MLE. And, for example, if you have some prior on the parameters, then you can use MAP as well. And that's used in many different Google products.

### As machine learning continues to mature, will MLE and MAP remain relevant?

I think no matter if you use like linear models or deep neural networks, the principle of Maximum Likelihood Estimate or Maximum Posterior still applies. So, these are the basic tools that we rely on to try to decide basically what kind of distribution the data will follow so that it decides on — I mean, even for the DNS, what kind of classifiers should sit on top of that. So, it's still very useful and we rely on that to decide on — like, for example, for predicting click-through rates, then you can use — if you

think of it, for Bernoulli distributions, and if you're trying to predict like, for example, the user's watch time, like a poisson classifier would make sense. So, these principles are still very useful — yeah, even if now people are relying more and more on these deep models instead of the linear models.

## Why is the Bayes Optimal Classifier useful to data scientists?

So from my understanding, the Bayes optimal classifier is based on an ensemble approach. So when it tries to make predictions, instead of traditionally relying on a single hypothesis that you got from either MLE or MAP — so what you're trying to do is, you do a weighted sum of all predictions using all of the possible hypotheses from your hypothesis class. And basically you try to weight each prediction by the kind of posterior distribution of each hypothesis. So I think it's really useful in the case where you have very limited data or you have noise in your data, which is — I guess it's true in any of the real-world applications. So, that makes it very useful. But at the same time it's usually not tractable to try to get the Bayes optimal classifier. So there's definitely a trade-off. In real-world applications you also need to cluster time and the resource, so there's a trade-off you need to make.

---

[Back to Table of Contents](#)

## Watch: Smoothing for Small Data Sets

Maximum Likelihood Estimation is very useful, particularly on larger data sets. But if your data set is small, it may be misleadingly skewed toward one label. The parameters that maximize the probability of the data can be extreme, and wrong. A technique to help reduce such inaccuracies induced by small data set sizes is to apply plus-one (+1) smoothing, otherwise known as Laplace smoothing. This involves inserting extra "hallucinated" samples into your data set based on prior knowledge or assumptions about the underlying distribution. Professor Weinberger explains more, and provides an example of how this technique of additive smoothing works.

### Video Transcript

Maximum Likelihood Estimation is a great way to estimate a distribution from data that's drawn from it. Especially when the data set becomes very large, it becomes very convenient and pretty accurate. However, it also has a downside, and that is when the data set is small. Let me walk through an example. So imagine we are estimating the probability of a coin coming up heads. Let's say we take this coin and we toss it six times. And believe it or not, six times it comes up heads; every single time. Well, what would be our MLE estimator of it coming up heads? Well, it will be the number of times heads divided by the total number of throws; that's 6 divided by 6 is 1. So we would estimate that this coin, this magical coin, has 100% chance of coming up heads and zero of coming up tails. Zero probability of coming up tails is of course nonsensical, right? It would mean that there's absolutely no way in this universe that it could come up tails. And that's impossible; we know that is wrong. So should we use that estimator when we know it's so terribly wrong? So if you know something about your data set, if you have some prior belief about what the distribution is, then you can inject that through hallucinated data, and we call that plus-one smoothing or Laplace smoothing. And the way this works is as follows: Imagine, for example, we have a coin toss, and coin tosses are typically 50-50. So, 50-50 would be that it comes up even number of heads, even number of tails. So, here is what would you do: Before you even look at the data, you start

out with some "hallucinated" data. You say, "I tossed the coin twice, and once it came up heads, once it came up tails." That hallucinated data would give rise to a probability of 50% heads and 50% tails. Now you observe the data. As you observe the data, the probability changes from 50-50 to something else. In this case, you see a drastic six times in a row it's being heads. That means after six rows, plus the two hallucinated rows, you now have seven heads over eight total tosses. That's the 88% chance of heads. That's still much higher than 50-50, but maybe that coin really is skewed. But certainly, you no longer believe that it's impossible to get tails. So plus-one smoothing is a very nice way of instilling additional information, prior information that you have that you know it can't possibly be the case that something never comes up. So what I'm doing is just adding in one set so that I'm sure I observe it at least once. If your prior belief is right, if the coin is really close to 50-50, this makes your convergence much, much, much faster; that means you need a lot less data to get a very good estimate of your distribution. Of course, the smoothing was as a downside; and that is, if you're, for example, wrong — let's say this is actually a coin that is extremely heavy on the face, on the head side, and comes up head 99% of the time. Well, in that case, by instilling one tail that I "hallucinated," I'm actually biasing towards a 50-50 and I have to observe more often that it comes up heads in order to converge to the true 99% probability of coming up heads. So, smoothing is a great way of instilling prior knowledge, and it's very effective if this prior knowledge is correct. But if it's incorrect, it can also hinder you.

---

[Back to Table of Contents](#)

## Activity: Test the Effect of Smoothing

Smoothing can have positive and negative effects on your probability calculation. If your prior belief is right, your calculation will likely be more accurate. However, if your prior belief is wrong, you could be injecting data into your calculation that unfavorably skews the result. The size of your data set also significantly impacts the influence your smoothing data has on the final probability calculation. Let's investigate this effect with a simple coin flipping activity.

### Instructions

**Flip a standard, unbiased coin ten times. Record the number of occurrence of both heads and tails.**

From your outcome estimate the probability of heads, using MLE. Use the smoothing technique of adding one head and one tail to your observed data to represent the prior belief of 50% heads. Recalculate after incorporating the new data from your prior belief. Do you think the smoothing has improved your estimate?

Now, instead of 50% heads as your prior belief, add two heads to your observed data to represent the prior belief of 100% heads. Recalculate after incorporating the new data from your prior belief. Do you think the smoothing has improved your estimate?

**Now flip a standard, unbiased coin thirty times. Record the number of occurrence of both heads and tails.**

Calculate the probability of heads, . Use the smoothing technique of adding one heads and one tails to your observed data to represent the prior belief of 50% heads. Recalculate after incorporating the new data from your prior belief. Do you think the smoothing has improved your estimate?

Now, instead of 50% heads as your prior belief, add two heads to your observed data to represent the prior belief of 100% heads. Recalculate after incorporating the new data from your prior belief. Do you think the smoothing has improved your estimate?

Did adding a prior belief have a larger impact on your probability calculation when the observed data was small? How did using an inaccurate prior belief skew your results?

---

[Back to Table of Contents](#)

## Module 1 Wrap-up: Estimating Distributions from Data

Estimating distributions from low-dimensional data is seemingly easy. In this module, you explored the Bayes optimal classifier and making the most accurate decisions possible when you know the distribution of your data. Since that is not usually the case in practice, you examined how maximum likelihood estimation offers a feasible way to estimate your data distribution. But, as you will soon find out, the curse of dimensionality will force us to revise our strategy yet again.

---

[Back to Table of Contents](#)

# Module 2: Classification with the Naive Bayes Algorithm

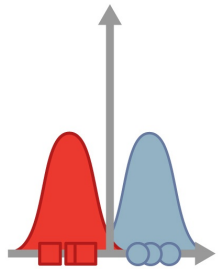
1. [Module 2 Introduction: Classification with the Naive Bayes Algorithm](#)
2. [Watch: Limitations of High Dimensional Space in MLE](#)
3. [Read: MLE Curse of Dimensionality](#)
4. [Read: Formalize the Curse of Dimensionality](#)
5. [Activity: Capturing All Possibilities in d Dimensions](#)
6. [Watch: Naive Bayes Assumption](#)
7. [Data Sets Where the Naive Bayes Assumption Holds](#)
8. [Watch: Naive Bayes Classifier](#)
9. [Read: Derivation of Naive Bayes Classifier](#)
10. [Tool: Naive Bayes Cheat Sheet](#)
11. [Watch: Determine Probability With Categorical Naive Bayes](#)
12. [Read: Categorical Naive Bayes Classifiers](#)
13. [Activity: Naive Bayes in Action](#)
14. [Module 2 Wrap-up: Classification with the Naive Bayes Algorithm](#)

---

[Back to Table of Contents](#)



## Module 2 Introduction: Classification with the Naive Bayes Algorithm



In this module, you will explore the Naive Bayes assumption and classifier. You will complete a brief activity in which you consider binary feature spaces. In a discussion related to the Naive Bayes assumption, you will examine examples of when the assumption will hold true and complete readings detailing the various approaches and applications of the Naive Bayes

algorithm. These concepts will help you build a baby name classifying system in the final module.

---

[Back to Table of Contents](#)

## Watch: Limitations of High Dimensional Space in MLE

MLE is great for estimating the distribution over the labels for a given input from its past occurrences. But what happens when you never see the same input twice? In this video, Professor Weinberger discusses the limitations of applying maximum likelihood estimation in high dimensional spaces.

---

[Back to Table of Contents](#)

## Read: MLE Curse of Dimensionality

**You can only construct probabilities for vectors whose exact combination of features have already been “seen” in the training set.**

**Directly estimating a Binomial or Bernoulli distribution with MLE is only useful if there are many training inputs with feature vectors identical to a test input.**

In the previous module, you saw that MLE can be used to estimate the parameters of a distribution from a data set. For the binomial distribution you can directly estimate the labels given an input. However, there is a major problem with this approach on highly dimensional data sets: the more dimensions in your data set, the less likely it becomes that any two feature vectors will match exactly. You can only construct probabilities for vectors whose exact combination of features have already been “seen” in the training set.

### Spam Email Example

Let’s suppose you are trying to build an email classifier. For the initial implementation of this email classifier, you’ll use a very simple set of three features:

1. Does the email contain the word “bacon”?
2. Does the email come from a recognized IP address?
3. Does the email contain any misspelled words?

Suppose the (very-limited) training data set looks like this, where each id

is an email, 0 represents “no”, 1 represents “yes”, and the label tells us if the email is spam or not.

ID	Contains word "bacon"	Recognized IP	Misspelled words	Label
1	0	0	1	spam
2	0	0	1	spam
3	1	0	0	spam
4	0	0	1	spam
5	1	0	1	spam
6	1	1	1	spam
7	0	0	1	spam
8	0	0	1	spam
9	1	0	1	spam
10	0	0	1	spam
11	0	1	0	not spam
12	0	1	1	not spam
13	1	1	1	not spam
14	0	1	1	not spam
15	0	1	0	not spam

[Click to check your knowledge of MLE.](#)

---

[Back to Table of Contents](#)

## Read: Formalize the Curse of Dimensionality

**Fitting a Binomial or Bernoulli distribution on the data with MLE is only useful if there are many training inputs with feature vectors identical to a test input.**

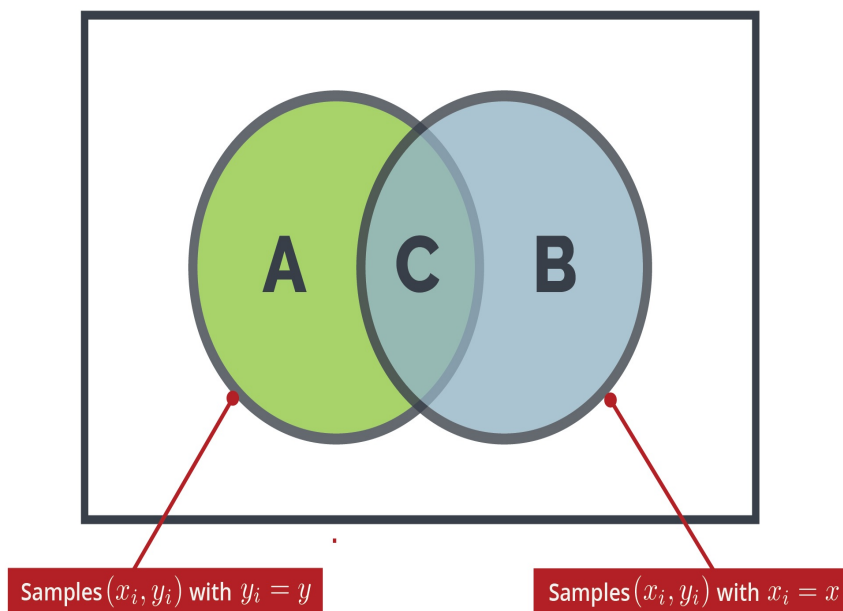
**Adding additional data to compensate for the lack of identical training and test points is not feasible in high dimensional spaces.**

Even in a very limited 3-dimensional case such as the previous spam email example, you notice that we cannot simply fit a Binomial Distribution to predict the label for each input on the data: **this approach is only possible if there are many training inputs with identical feature vectors of a test input.** You could try to collect more data to make sure you capture every possible feature vector in training, but consider the feasibility of such an approach as the dimensionality of the data set increases. As you include additional features in the classifier, such as other words that might indicate spam or metadata about the email itself, the possibility of two inputs matching exactly becomes more and more unlikely. Could you imagine exactly matching two inputs when you have hundreds or even thousands of different features?

When you fit a Binomial distribution to estimate the probability of discrete labels (i.e. spam or not spam) given an input  $\mathbf{x}$  i.e.  $P(y|\mathbf{x})$ , you use the following formula:

$$P(y|\mathbf{x}) = \frac{\sum_{i=1}^n I(\mathbf{x}_i = \mathbf{x} \wedge y_i = y)}{\sum_{i=1}^n I(\mathbf{x}_i = \mathbf{x})}$$

The numerator of the conditional probability expression is counting the number of times  $y_i$  takes on the value  $y$  *and* the remaining features  $x_i$  take on the value  $x$  (this is indicated by the symbol  $\wedge$  ).



Take a look at the Venn diagram. When you estimate the probability of a particular label given an input,  $P(y|x)$ , what you're effectively estimating is:

$$P(y|\mathbf{x}) = \frac{|C|}{|B|}$$

---

In high dimensional spaces, your data points become more "spread out" and it becomes less likely to observe any given feature value;  $|B| \rightarrow 0$  and  $|C| \rightarrow 0$ . But what is a high dimensional space? How quickly does this approach break down as you add more features to the vector?

---

[Back to Table of Contents](#)



## Activity: Capturing All Possibilities in d Dimensions

Suppose you have two binary features (i.e. your feature can only be one of two values, such as "0" or "1", "yes" or "no"). Now, what is the size of your feature space or equivalently, how many different combinations of features are possible?

**When you think you know the answer, click the image to reveal the solution**

*Click to reveal the answer*

Answer:  $2^2 = 4$

Now consider instead of two binary features, imagine you have twenty binary features. What is the size of your feature space?

*Click to reveal the answer*

Answer:  $2^{20} = 1,048,576$

With only twenty features, you already have a large input space, as it grows exponentially with the number of dimensions. In real data sets, we often have more than just twenty binary features (often we have thousands of features) and collecting all the data in the feature space is too expensive and not feasible. Soon you could obtain more possible inputs than there are electrons in the entire universe — certainly at this stage you would rarely (never) observe the same input more than once.

So how do you overcome the curse of dimensionality? How can we apply this methodology if it fails to work on common data sets in practice? To get around this problem, we will have to make a “naive” assumption to simplify our setting: the Naive Bayes assumption.

---

[Back to Table of Contents](#)

## Watch: Naive Bayes Assumption

Typically, the features of most data sets are not conditionally independent given the label. However, if we make that assumption anyway, we can greatly simplify our estimation approach. Concretely, we can turn a  $d$ -dimensional estimation problem, into  $d$  independent 1-dimensional estimation problems, each of which is easy to do. In this video, Professor Weinberger explains more about how this assumption works.

---

[Back to Table of Contents](#)

# Data Sets Where the Naive Bayes Assumption Holds

## Discussion topic:

The Naive Bayes assumption assumes that feature values are independent given the label. This is a very bold assumption. For this discussion, think of an example where this assumption would and would not hold.

Present an example scenario where the Naive Bayes assumption would or would not hold.

Explain why the assumption would or would not hold.

For example, the Naive Bayes classifier is often used in spam filtering, where the data is emails and the label is spam or not-spam. The Naive Bayes assumption implies that the words in an email are conditionally independent, given that you know that an email is spam or not. Clearly this is not true: we know, for example, that a spam email containing a word like "sale" is much more likely to contain a word like "price". More generally, neither the words in spam or not-spam emails are drawn independently at random, since we don't randomly choose words when constructing sentences in natural languages.

## Instructions:

You are required to participate meaningfully in all course discussions. The instructor will review the discussion and you will receive a "complete" or "incomplete" grade based on the quality of your contributions.

Limit your comments to 200 words.

You are strongly encouraged to post a response to at least two of your peers' posts.

## To participate in this discussion:

Click **Reply** to post a comment or reply to another comment. Please consider that this is a professional forum; courtesy and professional

language and tone are expected. Before posting, please review eCornell's policy regarding **plagiarism** (Links to an external site.)Links to an external site. (the presentation of someone else's work as your own without source credit).

---

[Back to Table of Contents](#)

## Watch: Naive Bayes Classifier

The Naive Bayes classifier utilizes the Naive Bayes assumption to estimate  $\theta$  from the training data and then predicts the most likely label for a given test input. In this video, Professor Weinberger discusses the underlying math behind the Naive Bayes classifier and explains how we can deal with the limitations of computing with finite precision.

---

[Back to Table of Contents](#)

## Read: Derivation of Naive Bayes Classifier

First you need to calculate the probabilities of the labels given the observed data, using Bayes' Rule.

Then, you choose  $y$  such that this probability is maximized, considering each dimension of the feature vector independently.

The Naive Bayes classifier decomposes a single high dimensional probability estimation problem into many 1-dimensional estimation problems by assuming the features are conditionally independent given the class label. We want to estimate  $\hat{P}(y|\mathbf{x})$  with the Naive Bayes classifier. To do so, we will make use of **Bayes' Rule and the Naive Bayes assumption**.

### Breaking Down Naive Bayes

You can estimate  $\hat{P}(y|\mathbf{x})$  by calculating  $P(y)$  and  $P(\mathbf{x}|y)$ , since by Bayes' Rule:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

Estimating  $P(y=c)$  (the probability that  $y$  takes on some value  $c$ ) is simple - it is

just the fraction of data points where  $y=c$  .

We can write this formally using an indicator variable  $I$ , which is 1 when the argument holds and 0 when it doesn't. If  $y$  takes on discrete binary values, such as 0 or 1, it simply counts how many times we observe each outcome:

$$P(y = c) = \frac{\sum_{i=1}^n I(y_i=c)}{n} = \hat{\pi}_c$$

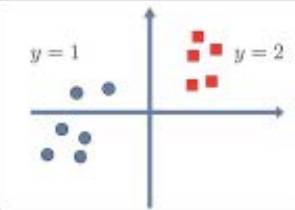
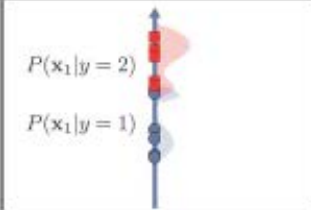
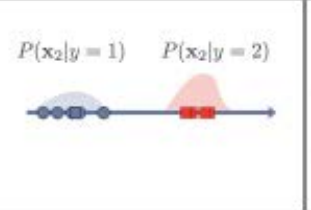
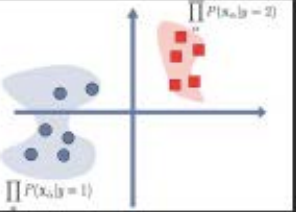
Estimating  $P(\mathbf{x}|y)$ , however, is not so simple. To do so, you will make use of the Naive Bayes assumption from earlier:

$$P(\mathbf{x}|y) = \prod_{\alpha=1}^d P(x_{\alpha}|y)$$

where  $x_{\alpha} = [\mathbf{x}]_{\alpha}$  is the value for feature  $\alpha$  .

Below, you'll see an illustration of how the Naive Bayes algorithm decomposes a multi-dimensional probability estimation problem (first image) into many 1-dimensional estimation problems. First, you estimate  $P(\mathbf{x}|y)$  independently in each dimension (center two images) and then obtain an estimate of the full data distribution by assuming conditional independence

$$P(\mathbf{x}|y) = \prod_{\alpha} P(x_{\alpha}|y) \quad \text{(rightmost image).}$$

Original	Estimation of first dimension	Estimation of first dimension	Resulting data distribution
			

## Derivation of Naive Bayes Classifier

Now, we can begin our derivation of the Naive Bayes classifier as follows:

$$\begin{aligned}
 h(\mathbf{x}) &= \operatorname{argmax}_y P(y|\mathbf{x}) \\
 &= \operatorname{argmax}_y \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} && \text{(Bayes rule)} \\
 &= \operatorname{argmax}_y P(\mathbf{x}|y)P(y) && (P(\mathbf{x}) \text{ does not depend on } y) \\
 &= \operatorname{argmax}_y \prod_{\alpha=1}^d P(x_{\alpha}|y)P(y) && \text{(by the Naive Bayes assumption)} \\
 &= \operatorname{argmax}_y \sum_{\alpha=1}^d \log(P(x_{\alpha}|y)) + \log(P(y)) && \text{(as log is a monotonic function)}
 \end{aligned}$$

Estimating  $P(x_{\alpha}|y)$  is easy since we only need to consider one single dimension, so MLE should work great. Estimating  $P(y)$  is not affected by the assumption and can similarly be done with straight-forward MLE.

---

[Back to Table of Contents](#)



## Tool: Naive Bayes Cheat Sheet

Use this [Naive Bayes Cheat Sheet](#) as a quick way to review the details of this classifier.

This tool provides an overview of the Naive Bayes classifier for your reference. You'll find information about the applicability, underlying mathematical principles, assumptions, and an example of the Naive Bayes classifier.

---

[Back to Table of Contents](#)

## Watch: Determine Probability With Categorical Naive Bayes

If we want to estimate distributions from data (for the sake of building a Naive Bayes classifier) we must make some assumptions. A simple assumption could be that our data was sampled from a binomial distribution, giving you Binomial Naive Bayes, also referred to as Categorical Naive Bayes. Professor Weinberger explains Categorical Naive Bayes further in the video below, and provides a basic example.

---

[Back to Table of Contents](#)

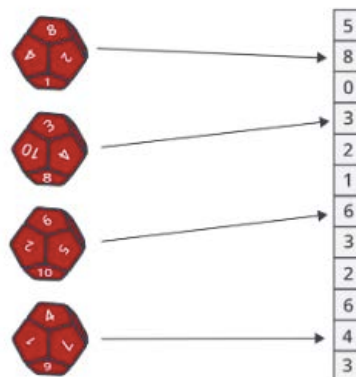
## Read: Categorical Naive Bayes Classifiers

For  $d$  dimensional data, imagine there exist  $d$  independent dice that represent each feature.

We assume training samples were generated by rolling one die after another, where there are  $K_{\alpha}$  possible values for each roll.

You've seen how to calculate  $P(y|x)$  using the Naive Bayes assumption and Bayes' Rule to calculate  $P(x|y)$ . However, depending on the types of features in your data, the way you calculate  $P(x|y)$  may differ. In this section, you will examine Naive Bayes when applied to categorical features, where each feature may fall into  $K_{\alpha}$  categories.

Below is a visualization of how categorical Naive Bayes works. For  $d$  dimensional data, imagine there exist  $d$  independent dice that represent each feature. We assume training samples were generated by rolling one die after another, where there are  $K_{\alpha}$  possible values for each roll. The value in dimension  $\alpha$  corresponds to the outcome that was rolled with the  $\alpha$ th die.



$$[\mathbf{x}]_{\alpha} \in \{f_1, f_2, \dots, f_{K_{\alpha}}\}$$

Per the expression above, the  $\alpha$ th feature is drawn from a set with  $K_{\alpha}$  elements. Each feature  $\alpha$  falls into one of  $K_{\alpha}$  categories. (Note that binary features are just a specific case of this, where  $K_{\alpha} = 2$ .) For example, you might have medical data where one feature could be "Does the patient have hypertension?" and the answer is binary (yes=1, no=0). In that case, we would have  $\mathbf{x}_{\alpha} \in \{0, 1\}$ .

Now we can begin to construct the model,  $P(\mathbf{x} | y)$ :

$$P(x_{\alpha} = j | y = c) = [\theta_{jc}]_{\alpha}$$

and

$$\sum_{j=1}^{K_{\alpha}} [\theta_{jc}]_{\alpha} = 1$$

In the equation above,  $[\theta_{jc}]_{\alpha}$  is shortcut notation denoting the probability that feature  $\alpha$  has the value  $j$ , given that the label is  $c$ . The second expression is simply a constraint that indicates that  $x_{\alpha}$  must fall into one of the categories  $\{1, \dots, K_{\alpha}\}$  i.e. the probabilities must sum to 1.

Now, we must estimate  $[\theta_{jc}]_{\alpha}$  itself to complete our model:

$$[\hat{\theta}_{jc}]_{\alpha} = \frac{\sum_{i=1}^n I(y_i = c) I(x_{i\alpha} = j) + l}{\sum_{i=1}^n I(y_i = c) + l K_{\alpha}}$$

where  $x_{i\alpha} = [\mathbf{x}_i]_{\alpha}$ ,  $l$  is a smoothing parameter and  $I$  is an indicator function

To train the Naive Bayes classifier, you must first estimate  $\theta_{jc}$  for all  $j$  and  $c$  and store them in the respective conditional probability tables (CPT). Also note that you can set the smoothing parameter to different values in order to use different estimation techniques:

- $l = 0$  is maximum likelihood estimation (MLE)
- $l > 0$  is maximum a posteriori (MAP)
- $l = +1$  is Laplace ("plus one") smoothing

In other words, without the  $l$  hallucinated samples, this formula means the probability that feature  $\alpha$  takes on value  $j$  given that the label  $c$  is:

$$[\hat{\theta}_{jc}]_{\alpha} = \frac{\# \text{ of samples with label } c \text{ that have feature } \alpha \text{ with value } j}{\# \text{ of samples with label } c}$$

Essentially, the categorical feature model associates a special die with each feature and label. Data is generated by first choosing the label (e.g. "healthy person"), which comes with a set of  $d$  dice, one for each dimension. The generator rolls each die, and fills in the feature value with the outcome of the die roll. So if there are  $C$  possible labels and  $d$  dimensions we are estimating  $dC$  dice from the data, but per the example only  $d$  of them are rolled. Die  $\alpha$  (for any label) has  $K_{\alpha}$  possible "sides". Of course, this is not how the data is generated in reality, but it is a modeling assumption that we make to approximate the real world.

We also need to estimate the probability of the class label independent of the labels:  $\hat{\pi}_c = P(y = c)$ . This denotes the probability that a sample is of label  $c$  without knowing anything about its features (e.g. out of the whole population, how many people have a particular illness, and how many don't). As this is just a simple one dimensional estimation problem, we can use MLE and compute the fraction of samples with label  $c$  out of the total set.

Putting all this together, we can formulate our Naive Bayes predictions for categorical features using Bayes Rule:

$$\operatorname{argmax}_y P(y = c \mid \vec{x}) \propto \operatorname{argmax}_y P(y = c) \prod_{\alpha=1}^d P(x_{\alpha} = j \mid y = c) = \operatorname{argmax}_y \hat{\pi}_c \prod_{\alpha=1}^d [\hat{\theta}_{jc}]_{\alpha}$$

## Activity: Naive Bayes in Action

Suzie is about to go on a blind date. Her friends want to help her to distinguish good from bad guys. Suzie's friends train a Naïve Bayes classifier to predict if someone is good or bad. As training data, they collect six people who they know are definitely good (Batman, Superman, Spiderman) or bad (Riddler, Penguin, Joker). Suzie's friends define the following features and labels:

Binary features:

**C** = "Guy wears a cape."

**M** = "Guy wears a mask."

**U** = "Guy wears his underwear outside his pants"

Labels:

**G** = "Guy is good"

**B** = "Guy is bad"

Features	Spiderman	Superman	Batman	Joker	Penguin	Riddler
Cape (C)	F	T	T	F	T	F
Mask (M)	T	T	F	F	F	T
Underwear outside of pants (U)	F	T	T	F	F	F

Label	G	G	G	B	B	B
-------	---	---	---	---	---	---

Recall these definitions from previous sections:

$$P(y = c) = \frac{\sum_{i=1}^n I(y_i = c)}{n} = \hat{\pi}_c$$

$$P(x = j|y = c) = \frac{\text{\# of samples with label } c \text{ that have feature } \alpha \text{ with value } j}{\text{\# of samples with label } c}$$

Click to check your knowledge of Naive Bayes in action.

Click for further Naive Bayes practice.

## Module 2 Wrap-up: Classification with the Naive Bayes Algorithm

In this module, you were introduced to the Naive Bayes assumption and classifier, and completed a brief activity considering feature spaces. You participated in a discussion related to the Naive Bayes assumption, along with relevant further readings on Naive Bayes classifiers. Now that you've examined the conceptual basis for Naive Bayes, it's time to put it into practice in Python.

---

[Back to Table of Contents](#)



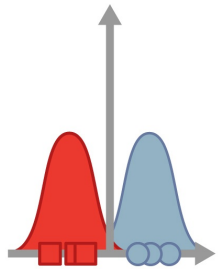
# Module 3: Building a Baby Name Classifying System

1. [Module 3 Introduction: Building a Baby Name Classifying System](#)
2. [Watch: Build a Baby Name Classifier](#)
3. [Watch: Determine Class Prior Probability](#)
4. [Code: Class Prior Probability](#)
5. [Watch: Determine Conditional Probability](#)
6. [Code: Conditional Probability](#)
7. [Watch: Create Feature Vectors from Data](#)
8. [Code: Feature Extraction and Hashing](#)
9. [Build a Baby Name Classification System](#)
10. [Improving Focus on Features](#)
11. [Module 3 Wrap-up: Building a Baby Name Classifying System](#)
12. [Read: Thank You and Farewell](#)
13. [Stay Connected](#)
14. [Stay Connected](#)

---

[Back to Table of Contents](#)

## Module 3 Introduction: Building a Baby Name Classifying System



Now that you are familiar with Naive Bayes and the Naive Bayes classifier, you are ready to begin applying the underlying principles to actual data. In this module, you will practice computing class and conditional probabilities. You will also have a chance to try feature extraction and hashing. Ultimately, you will build a baby name classifying system that can determine the gender of a name that you type in, based on certain features.

---

[Back to Table of Contents](#)

## Watch: Build a Baby Name Classifier

In this video, Professor Weinberger introduces your final project, where you will build a "baby name classifier." He explains how you will need to approach the problem and how Naive Bayes is applied to successfully determine the gender typically associated with a name.

---

[Back to Table of Contents](#)

## Watch: Determine Class Prior Probability

In this video, Professor Weinberger explains the first function you will implement in the project, "NaiveBayesPY," which simply determines the probability of a label given your training data set. In other words, prior to looking at features, NaiveBayesPY determines the probability a name drawn at random from your data will be a boy name or a girl name.

---

[Back to Table of Contents](#)

Code: Class Prior Probability

Code Check: What are the class priors?

Click to check your knowledge of class priors.

[Back to Table of Contents](#)

## Watch: Determine Conditional Probability

In this video, Professor Weinberger introduces the next function you will need to implement, `NaiveBayesPXY`, which predicts the probability of the input (name) given the label (boy or girl).

---

[Back to Table of Contents](#)

Code: Conditional Probability

Code Check: Product and Sum of Conditional Probabilities

Click to check your knowledge of conditional probability.



## Watch: Create Feature Vectors from Data

The features that describe characteristics of a name need to be stored in a way that makes them accessible by the classifier. In this video, Professor Weinberger explains how features can be turned into numbers through a process called hashing. You can then represent a name as a vector of zeros and ones with fixed dimensionality that describe the presence of the features.

---

[Back to Table of Contents](#)



## Code: Feature Extraction and Hashing

### Code Check: Feature Sum

Click to check your knowledge of feature extraction and hashing.

[Back to Table of Contents](#)

# Build a Baby Name Classification System

---

[Back to Table of Contents](#)

# Improving Focus on Features

## Discussion Topic:

What are some different ways to “featurize” text strings?  
In particular, what features seem to best capture gender in names?  
Can you come up with some better features than those used in the project?  
How well do these features apply across languages?  
Are the assumptions made in Naive Bayes reasonable for this application?

## Instructions:

You are required to participate meaningfully in all course discussions. The instructor will review the discussion and you will receive a "complete" or "incomplete" grade based on the quality of your contributions. Limit your comments to 200 words. You are strongly encouraged to post a response to at least two of your peers' posts.

## To participate in this discussion:

Click **Reply** to post a comment or reply to another comment. Please consider that this is a professional forum; courtesy and professional language and tone are expected. Before posting, please review eCornell's policy regarding plagiarism. Links to an external site. (the presentation of someone else's work as your own without source credit).

---

[Back to Table of Contents](#)

## Module 3 Wrap-up: Building a Baby Name Classifying System

In this module, you practiced computing probabilities and feature extraction. You completed a project that required you to use your conceptual and mathematical understanding in order to implement the Naive Bayes classifier to distinguish the typical gender associated with a baby name. Now that you've completed this module, you can extract features from text and apply Naive Bayes to classify any kinds of inputs within high-dimensional feature spaces.

---

[Back to Table of Contents](#)

## Read: Thank You and Farewell



**Kilian Weinberger**  
Associate Professor  
Computing and Information Science

Cornell University

Congratulations on completing "Estimating Probability Distributions."

In this class, you estimated probability distributions from data in order to make accurate predictions on unseen samples. It always astonishes me how some fairly technical mathematical derivations can lead to just a few lines of code in Python — that can do amazing things in the real world. Having finished all the assignments, you can pat yourself on the back: You have taken a big and important step towards becoming a true data scientist.

From all of us at Cornell University and eCornell, thank you for participating in this course.

Sincerely,

Kilian Weinberger

---

[Back to Table of Contents](#)

# Glossary

Download the [glossary](#) for this course.

---

[Back to Table of Contents](#)