

Probability for

HACKERS



Nipun Advilkar
Pycon India 2017

< About me >

- Machine Learning Engineer @Juxt-Smart Mandate
- AI and ML Enthusiast
- Likes to crack puns- Ni..pun (^-^)
- @nipunsadvilkar on GitHub
- More on website:

<https://nipunsadvilkar.github.io/>

#Questions:

1. How many of you are from heavy mathematical background?
E.g. Engineering, Physics
2. How many of you have used ML libraries like sklearn in your work?
3. How many of you want to get into following fields?
 - Artificial Intelligence
 - Machine Learning
 - Deep Learning
 - Data Science

MOTIVATION #1



CS229 Machine Learning Autumn 2016

Course Information

Instructors:

Andrew Ng, John Duchi

Course Description

This course provides a broad introduction to machine learning and statistical learning (generative/discriminative learning, parametric/non-parametric learning, neural networks, dimensionality reduction, kernel methods); learning theory (bias/variance tradeoff, PAC learning, VC theory, control). The course will also discuss recent applications of machine learning, including bioinformatics, speech recognition, and text and web data processing.

Prerequisites

Students are expected to have the following background:

- Knowledge of basic computer science principles and skills, at a level sufficient for CS229.
- Familiarity with the probability theory. (CS 109 or STATS 116)
- Familiarity with linear algebra (any one of Math 104, Math 113, or CS 209).

Prerequisite for any famous
AI and ML course/Book

Deep Learning

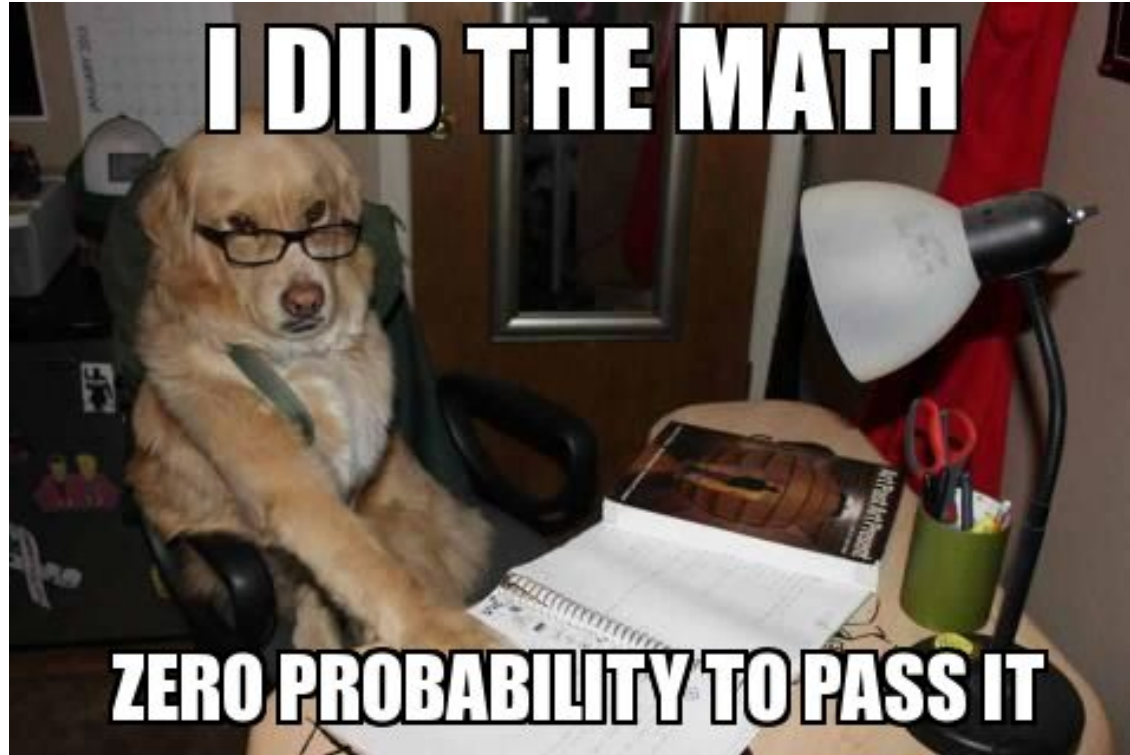
An MIT Press book

Ian Goodfellow and Yoshua Bengio and Aaron Courville

Deep Learning

- [Table of Contents](#)
- [Acknowledgements](#)
- [Notation](#)
- [1 Introduction](#)
- [Part I: Applied Math and Machine Learning Basics](#)
 - [2 Linear Algebra](#)
 - [3 Probability and Information Theory](#)
 - [4 Numerical Computation](#)
 - [5 Machine Learning Basics](#)
- [Part II: Modern Practical Deep Networks](#)

Those who struggle with math be like:





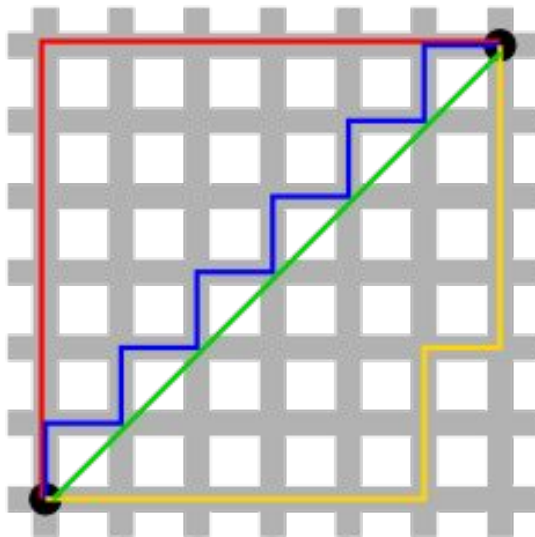
Hackers' approach to learn Math

- Math is difficult but through coding, we can make it more interactive and intuitive.
- I like this quote:
 - “Statistics is **Hard**.
Using programming skills it can be **easy**”
 - Jake VanderPlas (*Statistics for Hackers* - **Pycon 2016**)
[Same for Probability]
- Though, I want you to focus more on concepts and not on code (Code is available on GitHub. Have a look at it later)

MOTIVATION #2

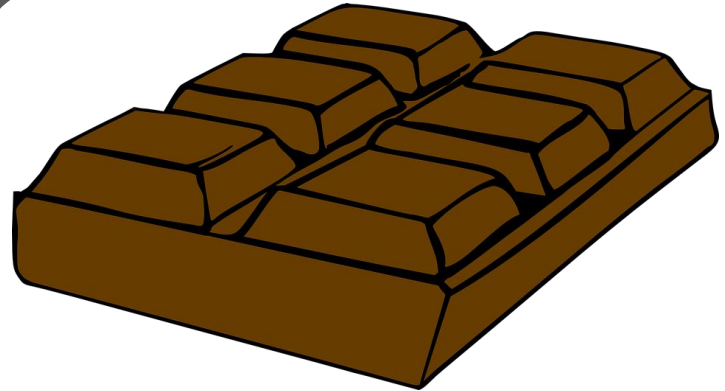
Modern AI

"Study and design of any agent that behaves in an intelligent way"



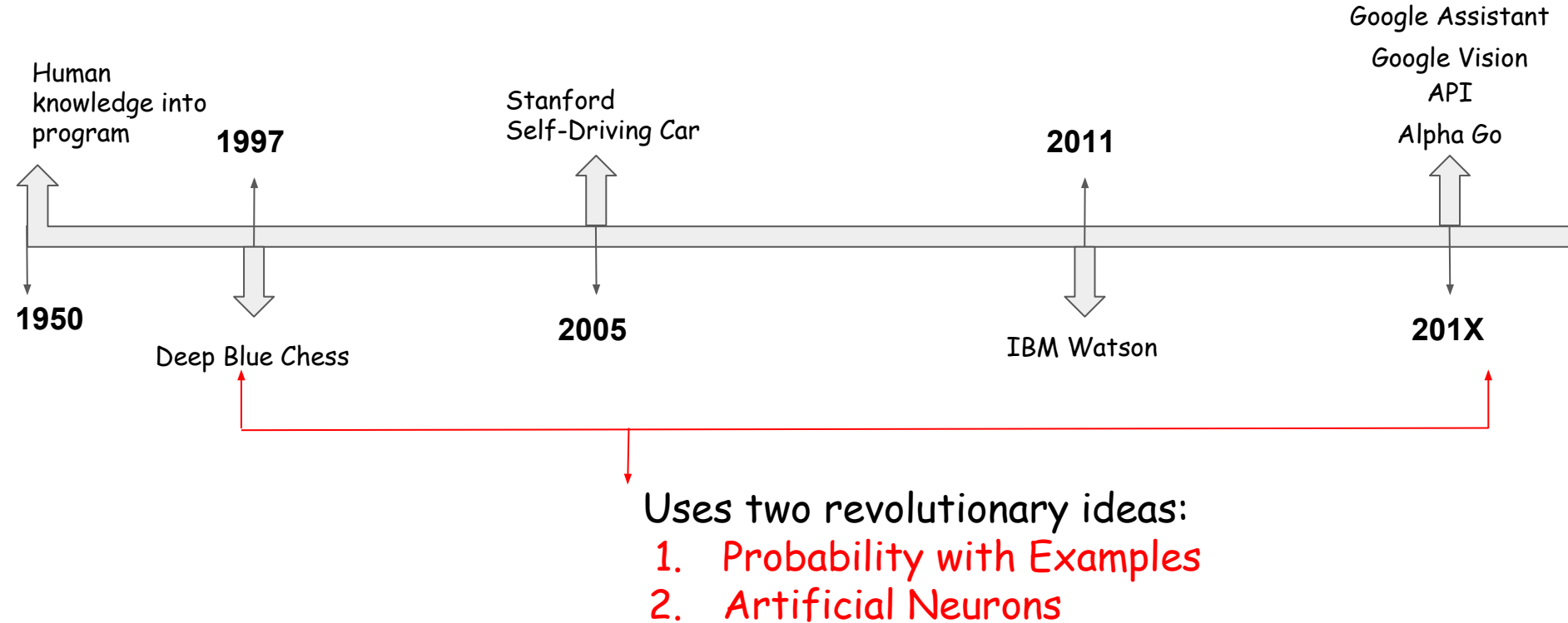
Demo Time!🕒

WHO LIKES CHOCOLATES
HERE?

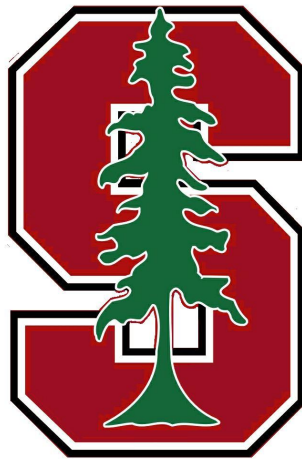
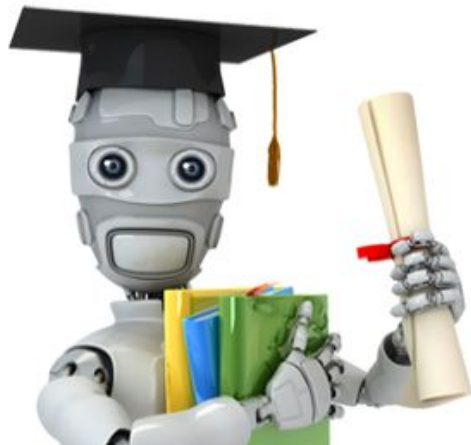


MOTIVATION #2

HISTORY OF AI



Conclusion



"Not once, but twice AI was revolutionized by people who understood Probability Theory"

- *Stanford University | CS 109: Probability for Computer Scientists*

TARGET

To be able to understand following math

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k)$$

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k) \prod_{i=1}^n p(x_i | C_k).$$

"Naive" conditional
independence assumptions

Diving into Probability

Obligatory coin toss experiment (interactive way)



Using:

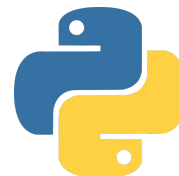
1. Virtual Coin with



2. Comparing theoretical to experimental probability with



3. Simulating experiment with Python



Link to Ipython Notebook:

[Introduction to Probability.ipynb](#)

#Activity 1 - Virtual Coin with



FLIP THE COIN

CHOOSE YOUR FAVOURITE COIN:

[MINE IS ₹10]



Link to Ipython Notebook:
[Introduction to Probability.ipynb](#)

#Activity 2 - Comparing theoretical to experimental probability with



Chance Events

Randomness is all around us. Probability theory is a mathematical framework that allows us to analyze chance events in a logically sound manner. The probability of an event is a number indicating how likely that event will occur. This number is always between 0 and 1, where 0 indicates impossibility and 1 indicates certainty. A classic example of a random experiment is a fair coin toss, in which the two possible outcomes are heads or tails. In this case, the probability of flipping a head or a tail is $1/2$. In an actual series of coin tosses, however, we may get more or less than exactly 50% heads.

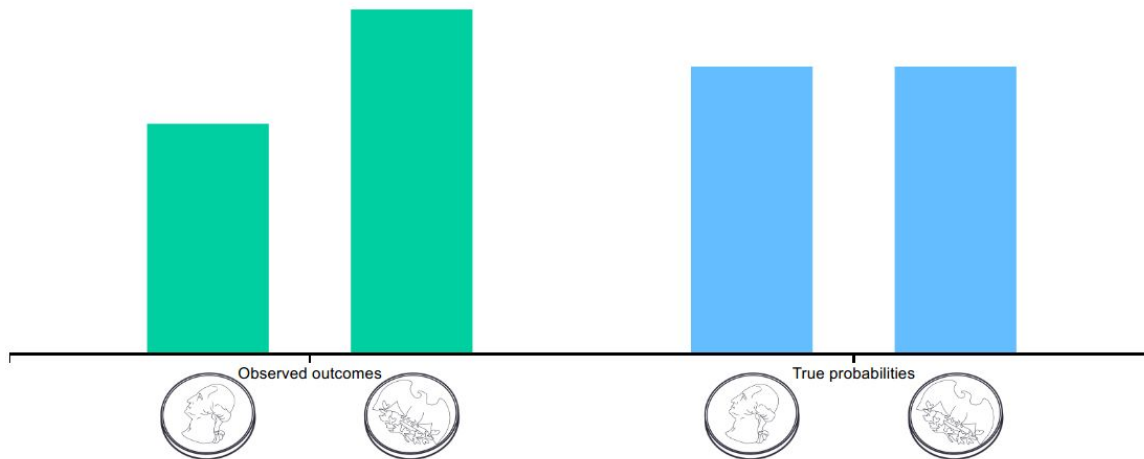


Flip the Coin

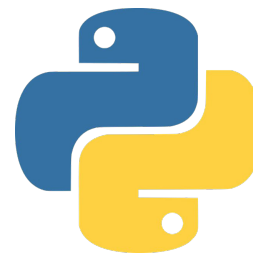
Flip 100 times

For an unfair or weighted coin, the two outcomes are not equally likely. You can change the weight of the coin by dragging the true probability bars up or down.

Link to Ipython Notebook:
[Introduction to Probability.ipynb](#)



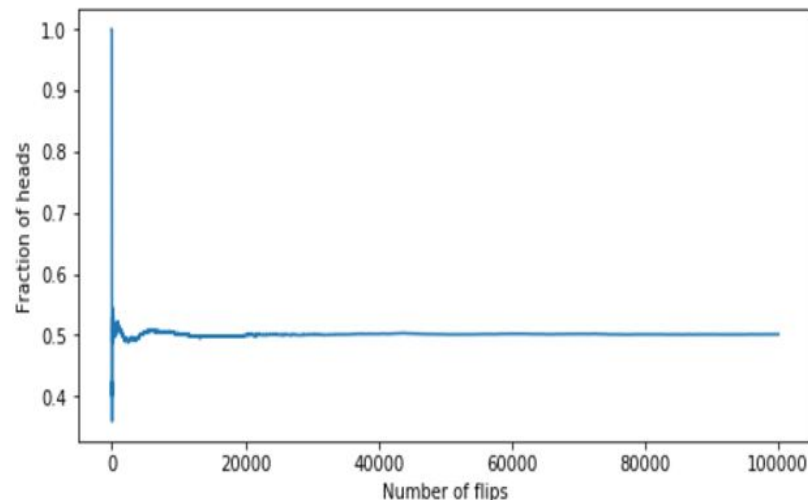
#Activity 3 - Simulating coin-toss experiment with Python



For number of coin-toss (n) = 100000

```
In [17]: n = 100000
heads_so_far = 0
fraction_of_heads = []
for i in range(n):
    if comp_prob_inference.flip_fair_coin() == 'heads':
        heads_so_far += 1
    fraction_of_heads.append(heads_so_far / (i+1))
```

```
In [18]: plt.figure(figsize=(8, 4))
plt.plot(range(1, n+1), fraction_of_heads)
plt.xlabel('Number of flips')
plt.ylabel('Fraction of heads')
```



Link to Ipython Notebook:
[Introduction to Probability.ipynb](#)

Ingredients to Modelling Uncertainty

1. Sample space:

The set of all possible outcomes for the experiment

$$\Omega = \{\text{heads}, \text{tails}\}$$

2. The probability of each outcome:

For each possible outcome, assign a probability that is at least 0 and at most 1. For the fair coin flip:


$$\mathbb{P}(\text{heads}) = \frac{1}{2} \text{ and } \mathbb{P}(\text{tails}) = \frac{1}{2}$$

Introduction to Random Variables

*Random
Variable*

*Possible
Values*

*Random
Events*

$$X = \begin{cases} 0 \\ 1 \end{cases}$$


Link to Ipython Notebook:
[Introducing Random Variables.ipynb](#)

Relation between Random Variables

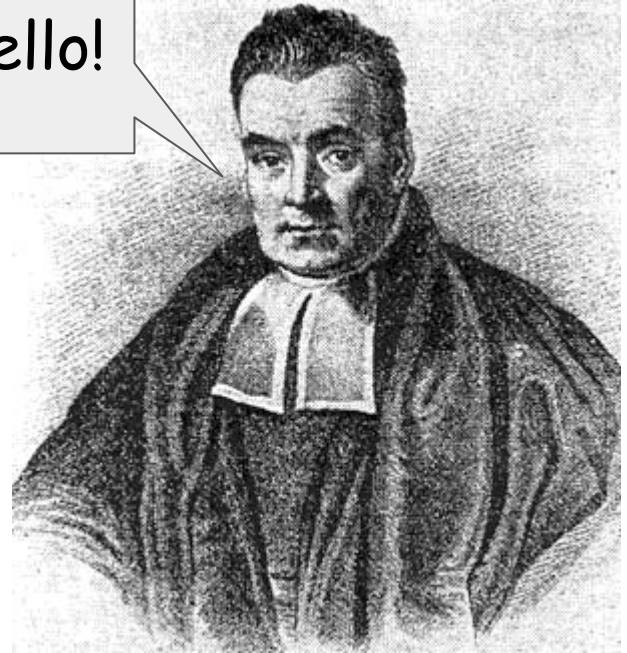
1. Joint Probability
2. Marginal Probability
3. Conditional Probability
4. Dependence & Independence

Link to Ipython Notebook:
[Relations between Random Variables.ipynb](#)

Demystifying Bayes theorem

$$\text{Posterior } P(A | B) = \frac{\text{Likelihood } P(B | A) \text{ Prior } P(A)}{\text{Evidence } P(B)}$$

Hello!



Application of Probability Theory learnt so far in Machine learning

Naive Bayes
Algorithm
As a Spam filter



Naive Bayes = Bayes theorem + Chain rule + Naive Conditional Independence

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

Denominator doesn't matter since it doesn't depend on the class

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

Chain Rule

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k)$$

"Naive" conditional independence assumptions

Take Away:

- Mathematics is **Hard**, using programming skills it can be **easy**
- Intuition of Probability theory behind simple yet fast Naive Bayes algorithm

~ Thank you! ~



<https://nipunsadvilkar.github.io>



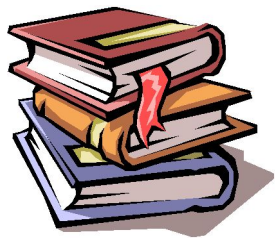
<https://github.com/nipunsadvilkar>



<https://facebook.com/nipunsadvilkar>



nipunsadvilkar@gmail.com



References:

1. [MITx: 6.008.1x Computational Probability and Inference, Edx](#)
2. [CS 109: Probability for Computer Scientists, Stanford University](#)
3. [Prob140: Probability for Data Science, UC Berkeley](#)
4. [Mathsisfun.com](#)
5. [\[https://en.wikipedia.org/wiki/Naive_Bayes_classifier\]\(https://en.wikipedia.org/wiki/Naive_Bayes_classifier\)](#)
6. [Seeing Theory, Brown University](#)
7. [<https://github.com/wintersummermint/coin-flip-javascript>](#)