



Programming meets Mathematics: **Probability**

Sunglok Choi, Assistant Professor, Ph.D.
Computer Science and Engineering Department, SeoulTech
sunglok@seoultech.ac.kr | <https://mint-lab.github.io/>

Programming meets Mathematics

~~Calculus~~ **Differentiation**

~~Linear Algebra~~ **Vector and Matrix**

~~Optimization~~ **Nonlinear Optimization** (as local optimization)

- Gradient descent: Selecting the search direction **with the 1st derivative**

- Possible problems: Too small and large step size

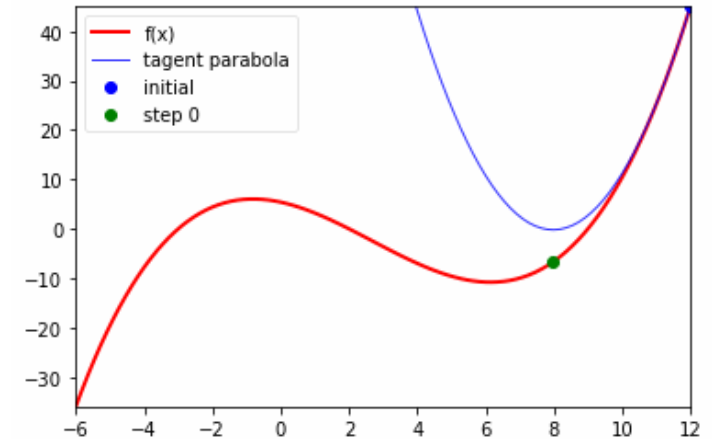
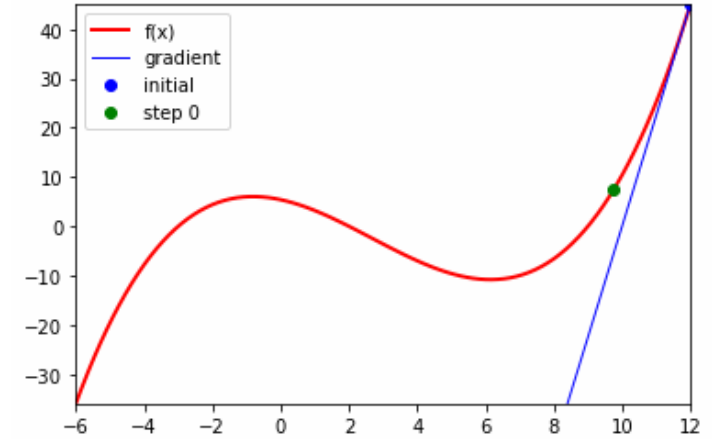
- Newton's method: Selecting the search direction and step **with the 1st and 2nd derivatives**

- Possible problems: The maxima problem, the saddle point problem

- [scipy.optimize](#): A magic wand **without derivatives**

▪ **Probability**

▪ **Information Theory**



Probability

- **Why probability?**

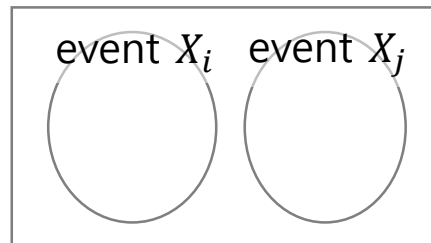
- Uncertain observation (\because noise and error)
- Incomplete data (\because unobservable or missing elements)
- Imperfect knowledge and rules/models (\because over-simplified or incorrect)

- **Probability**

- A numerical description of how likely an event is to occur or how likely a proposition is true
- Notation: $P(X)$ or $Pr(X)$ for the probability of an event X
- Axioms

1. For any event X , $0 \leq P(X)$
2. Probability of the sample space S is $P(S) = 1$
3. If X_1, X_2, \dots are disjoint events, then $P(X_1 \cup X_2 \cup \dots) = P(X_1) + P(X_2) + \dots$

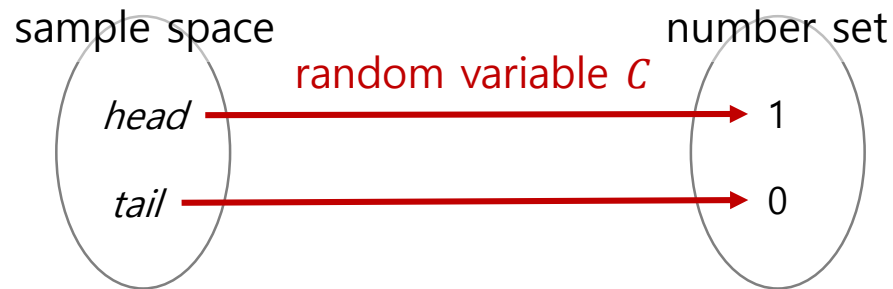
(mutually exclusive; $P(X_i \cap X_j) = 0$)



Probability

- Random variable

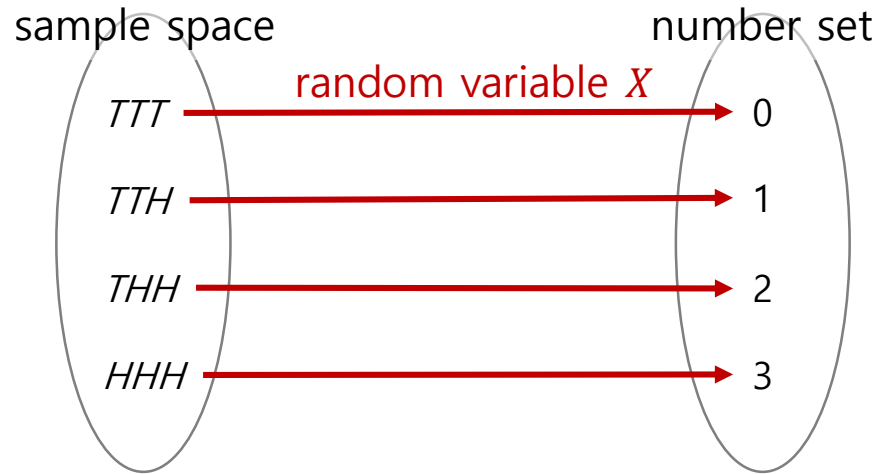
- Roughly, a mapping from a sample space to a measurable number set (usually \mathbb{R} or \mathbb{N})
- e.g. Event D : Rolling a dice (sample space: 1, 2, 3, 4, 5, and 6)
 - The probability of an event of rolling a dice: $P(D)$
 - The probability of getting 3 after rolling a dice: $P(D = 3)$
- e.g. Event H : Measure height of a student (sample space: \mathbb{R})
 - The probability of a student's height is more than 180: $P(H > 180)$
- e.g. Event C : Tossing a coin (sample space: *head* and *tail*)
 - The probability of getting *head* after tossing a coin: $P(C = 1)$



Probability

- Random variable

- Roughly, a mapping from a sample space to a measurable number set (usually \mathbb{R} or \mathbb{N})
- e.g. Event X : Tossing three coins together (sample space: TTT, TTH, THH, HHH)
 - The probability of getting two *head* after tossing the coins: $P(X = 2)$ if X is the number of heads

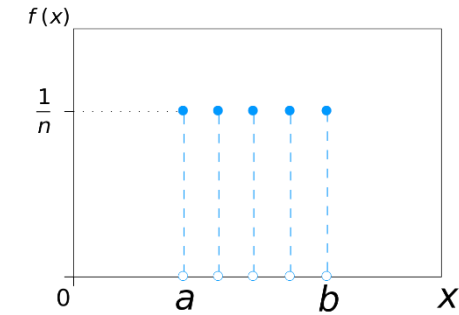


Probability

▪ Probability distribution

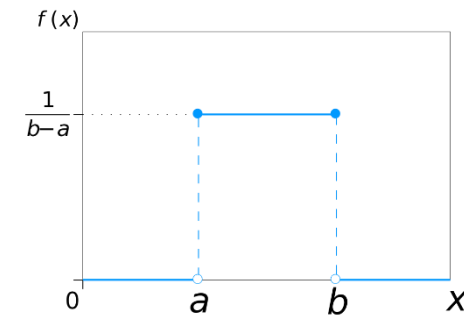
– [Probability mass function](#) (pmf) for **discrete** random variable

- $P(X = x_i) = p(X = x_i)$ (cf. In short, $p_X(x_i)$)
 - Each point has a probability value.
- From the axioms
 - $p_X(x_i) \geq 0$
 - $\sum_{x_i \in S_X} p_X(x_i) = 1$



– [Probability density function](#) (pdf) for **continuous** random variables

- $P(a \leq X \leq b) = \int_a^b f(X = x)dx$ (cf. In short, $f_X(x)$)
 - The area is a probability value, not a point.
- From the axioms
 - $f_X(x_i) \geq 0$
 - $\int_{-\infty}^{\infty} f_X(x) dx = 1$

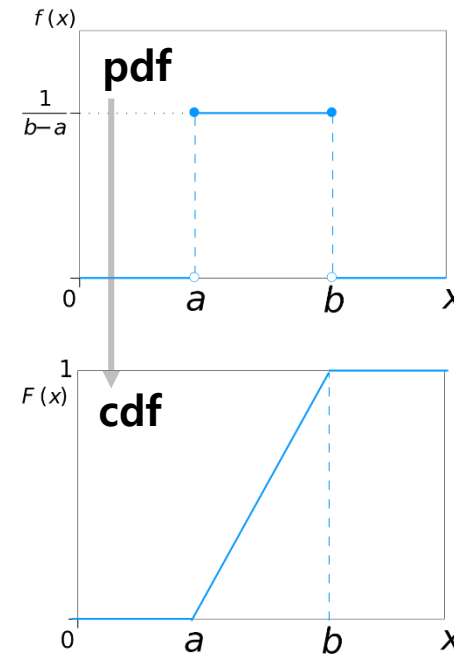
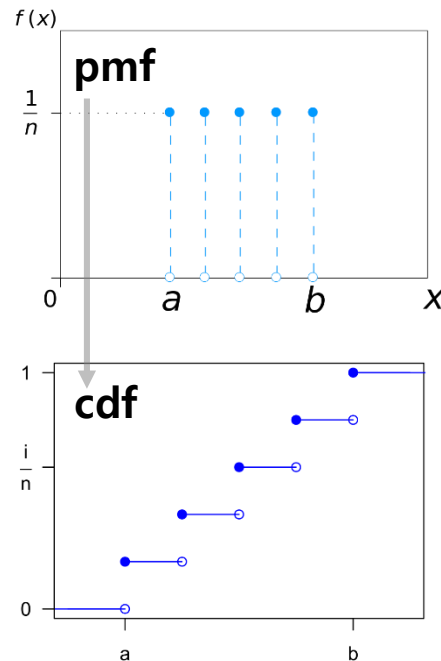


Probability

▪ Probability distribution

– Cumulative distribution function (cdf)

- $F_X(x) = P(X \leq x) \rightarrow P(a < X \leq b) = F_X(b) - F_X(a)$
- For a discrete random variable, $F_X(x) = \sum_{x_i \leq x} p_X(x_i)$
- For a continuous random variable, $F_X(x) = \int_{-\infty}^x f_X(t) dt$
- Properties: Non-decreasing, right-continuous, reaching 1 at the right end

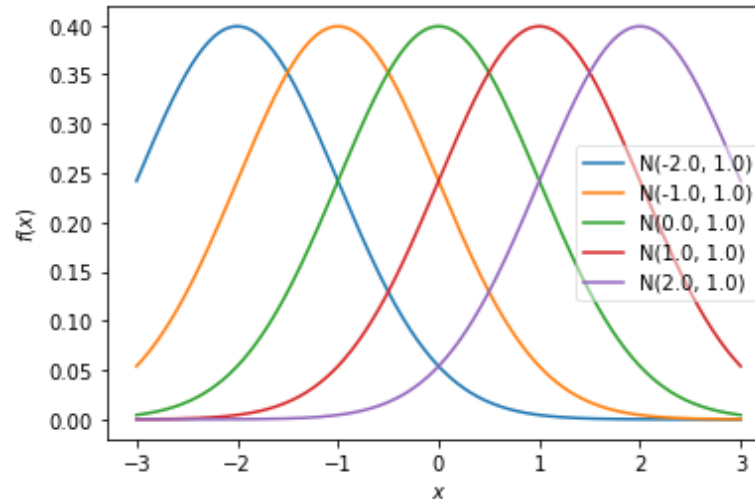
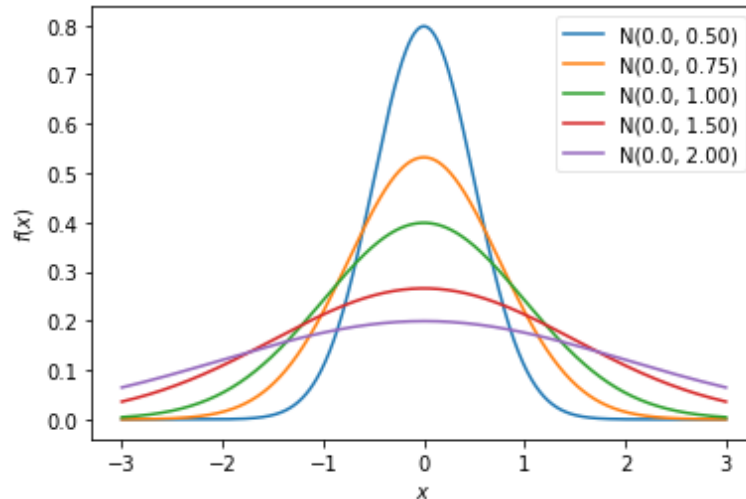
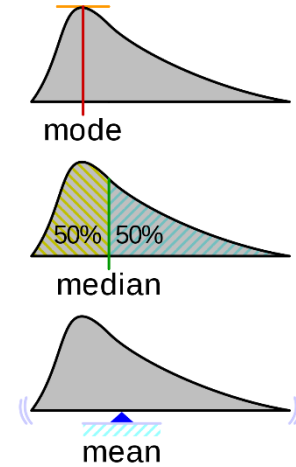


Probability

- **Probability distribution** [\[see more distributions\]](#)
 - e.g. [Normal distribution](#) (a.k.a. Gaussian distribution)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

- Notation: $N(\mu, \sigma^2)$
- Parameters: mean μ (~ location), variance σ^2 (~ squared width)
 - cf. mean = median = mode = μ
- Shapes (cf. $N(0, 1)$: standard normal distribution)



Probability

- **Probability distribution** [\[see more distributions\]](#)

- e.g. [Normal distribution](#) (a.k.a. Gaussian distribution)

- Example) Visualize the normal distribution with varying parameters

```
import numpy as np
import matplotlib.pyplot as plt

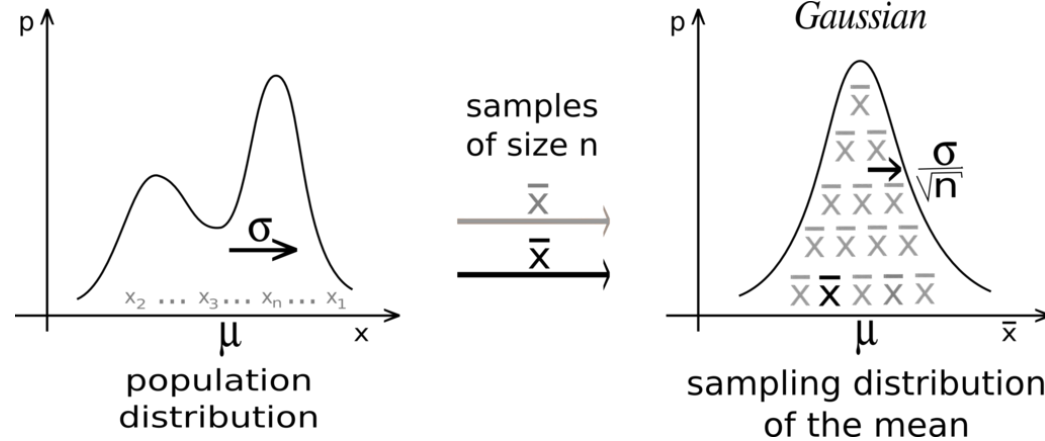
xs = np.linspace(-3, 3, 1000)
mu_set = [0] # [0] or [-2, -1, 0, 1, 2]
sigma_set = [0.5, 0.75, 1, 1.5, 2] # [1] or [0.5, 0.75, 1, 1.5, 2]

for mu in mu_set:
    for sigma in sigma_set:
        pdf = 1 / sigma / np.sqrt(2*np.pi) * np.exp(-0.5*((xs - mu)/sigma)**2)
        plt.plot(xs, pdf, label=f'N({mu:.1f}, {sigma:.2f})')

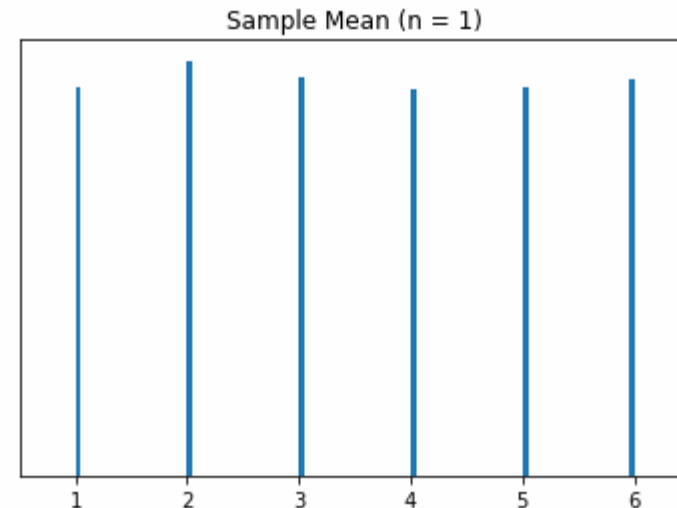
plt.xlabel('$x$')
plt.ylabel('$f(x)$')
plt.legend(framealpha=0.5)
plt.show()
```

Probability

- **Probability distribution** [\[see more distributions\]](#)
 - e.g. [Normal distribution](#) (a.k.a. Gaussian distribution)
 - Why important? The [central limit theorem](#) (CLT)



- Example) Validating CLT (1/2)
 - Rolling a dice (discrete uniform dist.)



Probability

- **Probability distribution** [\[see more distributions\]](#)
 - e.g. [Normal distribution](#) (a.k.a. Gaussian distribution)

- Example) Validating CLT (2/2)

```
import numpy as np
import matplotlib.pyplot as plt

hist_pts = 10000
hist_bins = 100
dice_range = (1, 6)
n = 100

# Acquire multiple sample means
samples = []
for i in range(hist_pts):
    samples.append(np.mean(np.random.randint(dice_range[0], dice_range[1]+1, n)))

# Visualize the distribution of sample means
plt.title(f'Sample Mean (n = {n})')
plt.hist(samples, bins=hist_bins, range=dice_range, density=True, align='mid')
plt.xlim(dice_range[0]-0.5, dice_range[1]+0.5)
plt.yticks([])
plt.show()
```

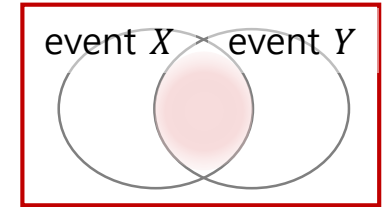
Central Limit Theorem



Probability

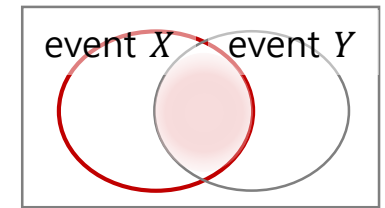
▪ Joint probability

- How likely would X and Y happen together? $P(X, Y)$ or $P(X \cap Y)$
- cf. Independence: $P(X, Y) = P(X) P(Y)$



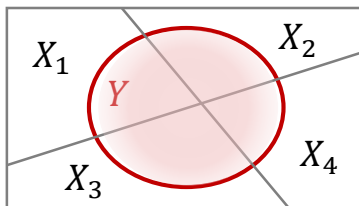
▪ Conditional probability

- Given X , how likely would Y happen? $P(Y|X) = \frac{P(Y, X)}{P(X)}$
- cf. Independence: $P(Y|X) = P(Y)$
- Chain rule: $P(Y, X) = P(Y|X) P(X) \rightarrow P(X_3, X_2, X_1) = P(X_3|X_2, X_1) P(X_2|X_1) P(X_1)$
- Bayes' theorem: $P(Y|X) = \frac{P(X|Y) P(Y)}{P(X)}$ (posterior, likelihood, prior, and marginalization)



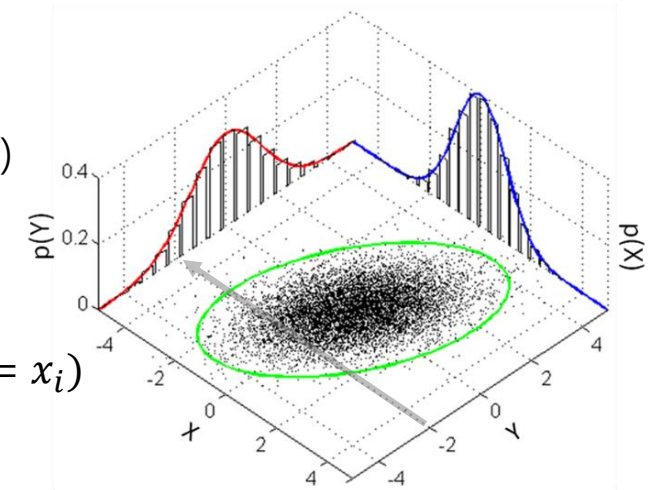
▪ Marginal probability

- Regardless of what happens to X , how likely would Y happen? $P(Y) = \sum_{x_i \in S_X} P(Y, X = x_i)$
- Law of total probability: $P(Y) = \sum_i P(Y, X_i) = \sum_i P(Y|X_i) P(X_i)$



if $\{X_i | i = 1, 2, \dots\}$ is a set of pairwise disjoint events

whose union is the entire sample space.



Probability

- Bayes' theorem

- $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$ (posterior, likelihood, prior, and marginalization)
- Alias: Bayes' law, Bayes' rule, and Bayesian theorem
- Example) Pancreatic cancer rate (췌장암 in Korean)
 - Patients with the cancer have a certain symptom: 100%
 - Occurrence rate of the cancer: 1 / 100,000
 - Occurrence rate of the same symptom for healthy persons: 10 / 100,000
 - **If you have the symptom, what is the probability that you have the cancer?**

Probability

- Bayes' theorem

- $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$ (posterior, likelihood, prior, and marginalization)

- Alias: Bayes' law, Bayes' rule, and Bayesian theorem

- Example) Pancreatic cancer rate (췌장암 in Korean)

- Patients with the cancer have a certain symptom: 100%

- Occurrence rate of the cancer: 1 / 100,000

- Occurrence rate of the same symptom for healthy persons: 10 / 100,000

- **If you have the symptom, what is the probability that you have the cancer?**

X

$P(Y|X)$

Y

- From the law of total probability, $P(X) = P(X|Y)P(Y) + P(X|\neg Y)P(\neg Y)$

- Therefore, $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{1}{11} \approx 9.1\%$

Symptom \ Cancer	Yes (Y)	No (¬Y)	Total
	Yes (X)	No (¬X)	Total
Yes (X)	1	10	11
No (¬X)	0	99989	99989
Total	1	99999	100000

If you know the joint probability distribution, you can derive anything you want.

→ $P(X|Y) = 1$

→ $P(Y) = 0.00001$, $P(\neg Y) = 0.99999$

→ $P(X|\neg Y) = 0.0001$

Probability

- Expectation

- A generalization of weighted average

- Alias: Mean, average, the **first moment**

$$E[X] = \sum_{i=1}^n x_i P(x_i) \quad \text{or} \quad E[X] = \int_{\mathbb{R}} x f(x) dx$$

- cf. Arithmetic mean $(\frac{\sum_{i=1}^n x_i}{n})$ is the expectation under uniform distribution.

- Properties

- Linearity: $E[X + Y] = E[X] + E[Y]$ and $E[aX] = aE[X]$
 - Non-multiplicativity: $E[XY] \neq E[X] E[Y]$
 - cf. If X and Y are independent, $E[XY] = E[X] E[Y]$

Probability

- Variance

- The expectation of the squared deviation of X from its mean
 - Alias: The **second central moment**

$$\text{Var}(X) = E[(X - \mu)^2]$$

- Calculation: $\text{Var}(X) = E[X^2] - E[X]^2 = E[X^2] - \mu^2$
- Properties
 - $\text{Var}(X) \geq 0$: Non-negative
 - $\text{Var}(X + a) = \text{Var}(X)$: Invariant to a location parameter
 - $\text{Var}(aX) = a^2 \text{Var}(X)$: Squared scale
 - $\text{Var}(aX + bY) = a^2 \text{Var}(X) + b^2 \text{Var}(Y) + 2ab \text{cov}(X, Y)$

Probability

- Variance

- The expectation of the squared deviation of X from its mean
 - Alias: The **second central moment**

$$\text{Var}(X) = E[(X - \mu)^2]$$

- cf. Covariance

- The joint variability of two or more random variables

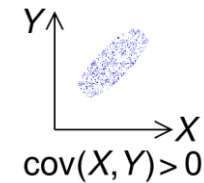
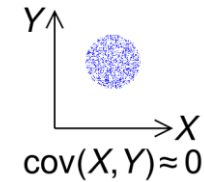
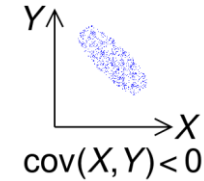
$$\text{cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

- A single variable: $\text{cov}(X, X) = E[(X - E[X])(X - E[X])] = \text{Var}(X)$

- cf. Correlation

- The normalized covariance (range: $[0, 1]$)
 - Alias: Dependence

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$



Probability

- [Variance](#), [Covariance](#), and [Correlation](#)

- Example) Correlation of the midterm and final exam scores

```
# Load score data
```

```
scores = np.array(read_data('data/class_score_*.csv'))
```

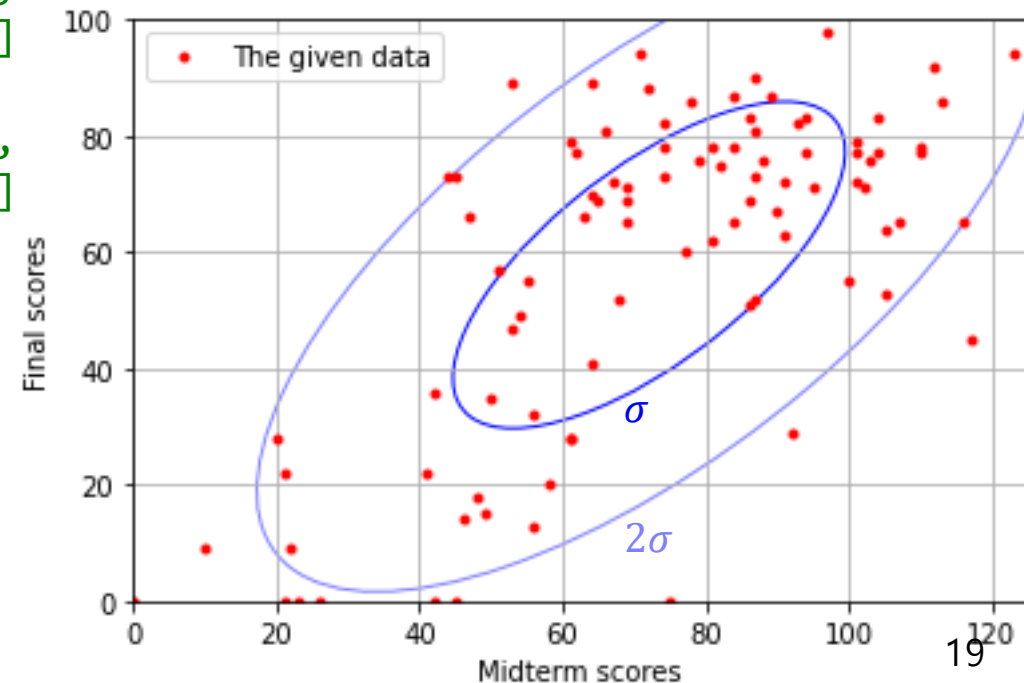
```
# Calculate the variance, covariance, and correlation
```

```
midtm = [func(scores[:,0]) for func in [np.mean, np.var, np.std]] # [72.07, 751.26, 27.41]
```

```
final = [func(scores[:,1]) for func in [np.mean, np.var, np.std]] # [57.81, 790.18, 28.11]
```

```
cov_all = np.cov(scores.T, ddof=0) # [[751.26, 534.94],  
print(cov_all)                    # [534.94, 790.18]]
```

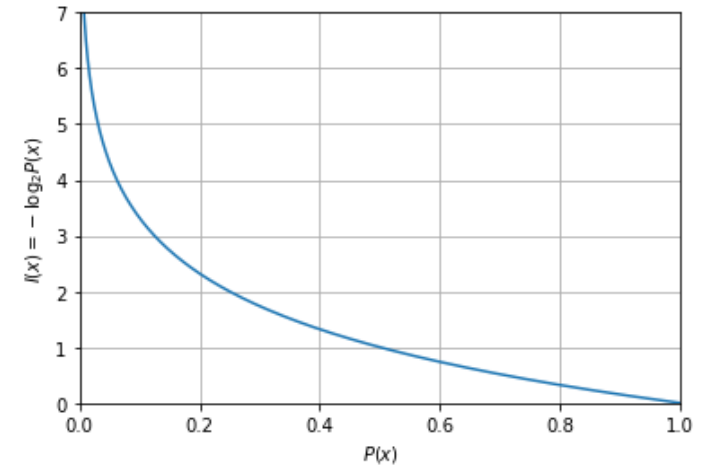
```
cor_all = np.corrcoef(scores.T)  # [[ 1.  ,  0.69],  
print(cor_all)                  # [ 0.69,  1.  ]]
```



Information Theory Cross Entropy

- (Shannon) Information of an event X

$$I(x) = -\log_2 P(x)$$



- An alternative way of expressing probability
 - Alias: Surprisal, information content
- As the concept of *surprisal*, Shannon information is roughly the level of surprise if the event is true (~ more surprises, less probable, more informative)
 - e.g. What is the most significant news if the news is correct?
 - 1) Tomorrow the sun will rise in the east.
 - 2) Tomorrow it will rain.
 - 3) Tomorrow the sun will rise in the west.
- In the view of compression or communication, Shannon information is the length of a message necessary for an optimal coding of the random variable
 - e.g. Shannon information of tossing two coins: Each probability $P(x) = \frac{1}{4} \rightarrow I(x) = 2$ (2-bit coding)
 - e.g. Shannon information of rolling a dice: Each probability $P(x) = \frac{1}{6} \rightarrow I(x) = 2.585$ (3-bit coding)

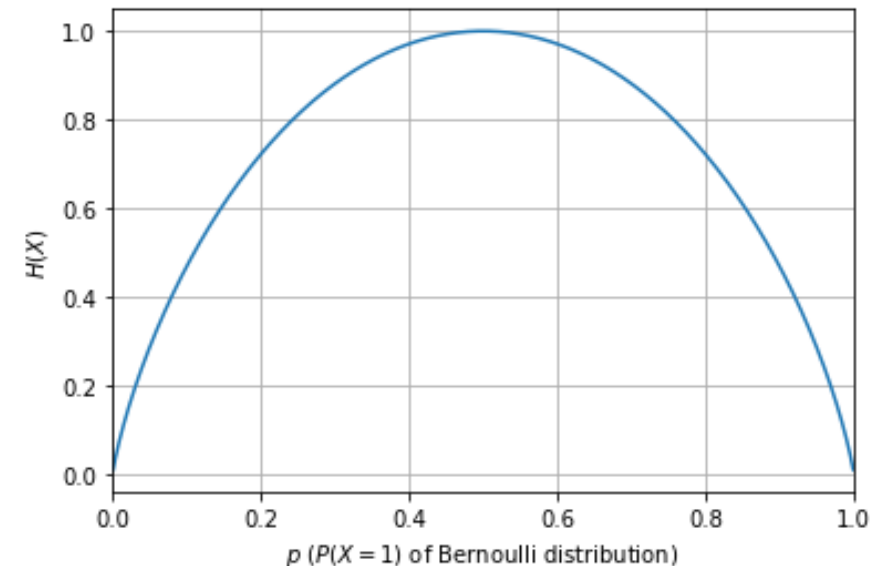
Cross Entropy

- [\(Shannon\) Entropy](#) of a random variable X

$$H(X) = E_X[-\log P(X)] = -\sum_{i=1}^n P(x_i) \log P(x_i)$$

- The expectation of Shannon information (or surprise or the number of bits with an optimal coding) inherent in the variable's all possible outcomes
- Example) Entropy of tossing an **unfair** coin (*head* = 1, *tail* = 0)
 - [Bernoulli distribution](#): $p(X = k) = \begin{cases} p & \text{if } k = 1 \\ 1 - p & \text{otherwise} \end{cases}$

```
p = np.linspace(0, 1, 1000)
entropy = -p*np.log2(p) - (1-p)*np.log2(1-p)
```



Cross Entropy

- (Shannon) Entropy of a random variable X

- Example) Entropy of rolling an **unfair** dice

- Non-uniform distribution

$$P_X(1) = P_X(2) = 0.4 \text{ and } P_X(3) = P_X(4) = P_X(5) = P_X(6) = 0.05$$

```
p = np.array([0.4, 0.4, 0.05, 0.05, 0.05, 0.05])
entropy = sum(-p*np.log2(p)) # 1.922...
```

- Q) How to encode them to achieve the average 2-bit message?

- A)

X	Code
1	0
2	10
3	11 00
4	11 01
5	11 10
6	11 11

Cross Entropy

- Cross entropy of the distribution q relative to a distribution p (over the same underlying sample space)

$$H(p, q) = E_p[-\log q] = - \sum_{x \in S_X} p(x) \log q(x)$$

- The average number of bits if a coding scheme is optimized for probability distribution q instead of the true distribution p
- Cross entropy roughly represents difference of two probability distributions similar to the [Kullback–Leibler divergence](#).
- Example) Cross entropy of rolling an **unfair** dice

```
p = np.array([1/6, 1/6, 1/6, 1/6, 1/6, 1/6])
entropy = sum(-p*np.log2(p)) # 2.585...
```

```
q1 = np.array([0.4, 0.4, 0.05, 0.05, 0.05, 0.05])
entropy1 = sum(-p*np.log2(q1)) # 3.322...
```

```
q2 = np.array([0.2, 0.2, 0.2, 0.2, 0.1, 0.1])
entropy2 = sum(-p*np.log2(q2)) # 2.655...
```

```
q3 = np.array([0.6, 0.2, 0.1, 0.05, 0.04, 0.01])
entropy3 = sum(-p*np.log2(q3)) # 3.665...
```

Summary

- **Probability**

- Why probability? What is probability?
- **Random variable** ~ Mapping from events to numbers
- Probability distribution
 - Representation: [pmf](#) (for discrete; ~ histogram), [pdf](#) (for continuous), [cdf](#) (for both)
 - e.g. [Normal distribution](#) (and the [central limit theorem](#))
- Joint probability, conditional probability, and marginal probability
 - [Chain rule](#), [Bayes' theorem](#), and [law of total probability](#)
- [Expectation](#), [variance](#), [covariance](#) (~ variance of two or more variables), and [correlation](#) (~ normalized covariance)

- ~~Information Theory~~ **Cross Entropy**

- Shannon information ~ The level of surprise (another representation of probability), but related to optimal coding
- Entropy ~ The expectation of Shannon information
- Cross entropy ~ Difference of two probability distributions