Python for ML

Modeling Uncertainty

Nadia Udler

In this lecture..

Mathematical models
Why model uncertainty
Deterministic variables vs random variables
Expected value of random variable
Continuous random variable– slides 3-4
Discrete random variable– slides 5-10
Morra Game revisited
Monte Carlo method for computing area– slides 11-14
Markov process
Bayes Rule – slide 35
MLE and MAP – slide 36
Naïve Bayes Classification – slides 39

Mathematical model: examples

A company is planning a corporate dinner in a park. There are 50 employees in the company. The dinner for 1 person costs $60 (including delivery cost). How much is the cost of the dinner for all employees?

Variables: c – total cost of the dinner
Answer: c = 50*60=3000

Mathematical model: examples

A company had a corporate dinner in a park. There are 50 employees in the company. The dinner for 1 person costs $60 (including delivery cost). Each employee could invite up to 4 guests. After the party the actual cost of the dinner was calculated to be $4500. What was the total number of guests?

Variables: total number of guests - g

$60(50+g) = 4500$ – mathematical model

$3000+60g = 4500$

$g = 1500/60 = 25$

g – deterministic variable

Why model uncertainty
Random Variables
Expected value of random variable

Real life example:

Stock price S – random variable
Today: S=S0
Tomorrow: S = S1 with probability p1, S = S2 with probability p2
Expected value E(S) = S1*p1+S2*p2

Game: throwing a die , gain the number shown on the side :
Gain G may be modeled as random variable
Possible values of G : 1,2,3,4,5,6, with equal probabilities.
Expected Value E(G) = (1+2+3+4+5+6)/6 = 3.5

Game: flipping a coin, gain 1 if heads occur, 0 otherwise
Gain G may be modeled as random variable
Possible values of G : 1 or 0 with equal probabilities.
Expected Value E(X) = (1+0)/2= 0.5

Examples of random variables with continuous distributions:

Buying lottery tickets every day. The time to first win T may be modeled as random variables that follows exponential distribution with parameter λ that has a meaning of winning rate.

Probability density function $p(x, \lambda) = \begin{cases} \lambda e^{-\lambda x}, x \geq 0 \\ 0, \ x < 0 \end{cases}$

The expected value of T is $E(X) = \int_0^\infty x\, p(x, \lambda) dx = \int_0^\infty x\, \lambda e^{-\lambda x} dx = \frac{1}{\lambda}$

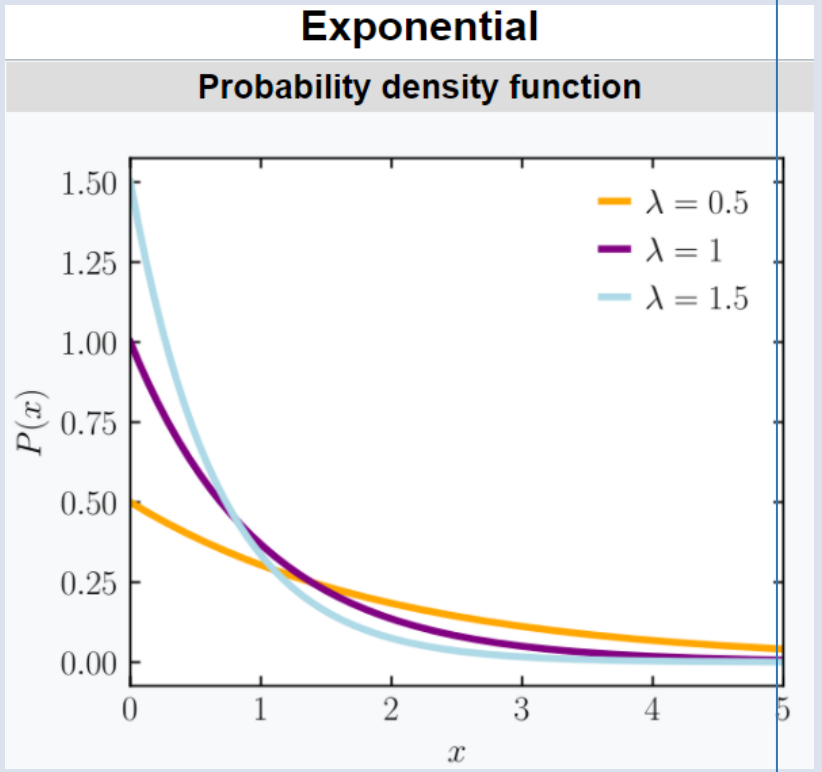
Time until first failure of the device. It may be modeled as random variable the exponential distribution with parameter λ which has a meaning of hazard rate.

Exponential distribution with parameter λ : finding expected value

$$E(X) = \int_0^{\infty} x\,\boldsymbol{\lambda e^{-\lambda x}}\,\boldsymbol{dx}$$

Using integration by parts formula

$$\int \boldsymbol{u dv} = \boldsymbol{uv} - \int \boldsymbol{v du}$$



$$\boldsymbol{u = x, dv = \lambda e^{-\lambda x}, v = \quad -e^{-\lambda x}}$$

$$E(X) = \int_0^{\infty} x\,\boldsymbol{\lambda e^{-\lambda x}}\,\boldsymbol{dx} = \quad \boldsymbol{-xe^{-\lambda x}}\,|_{\boldsymbol{0}}^{\infty} - \int_0^{\infty}\left(\boldsymbol{-e^{-\lambda x}}\right)dx =$$

$$= \boldsymbol{-xe^{-\lambda x}}\,|_{\boldsymbol{0}}^{\infty} - \frac{\boldsymbol{e^{-\lambda x}}}{\boldsymbol{\lambda}}\,|_{\boldsymbol{0}}^{\infty} = -\frac{\boldsymbol{1}}{\boldsymbol{e^{\lambda x}}}\left(\boldsymbol{x} + \frac{\boldsymbol{1}}{\boldsymbol{\lambda}}\right)|_{\boldsymbol{0}}^{\infty} = \frac{\boldsymbol{1}}{\boldsymbol{\lambda}}$$

Mathematical model: example1

A company is planning a corporate dinner in a park. There are 50 employees in the company. The dinner for 1 person costs $60 (including delivery cost). Each employee could invite up to 4 guests.
What is the expected cost of dinner for all? What is the maximum cost of dinner for all?

Mathematical model: example1

A company is planning a corporate dinner in a park. There are 50 employees in the company. The dinner for 1 person costs $60 (including delivery cost). Each employee could invite up to 4 guests.
What is the expected cost of dinner for all? What is the maximum cost of dinner for all?

Variables: number of guests of one employee - g
g – random variable which takes values 0,1,2,3,4 with equal probabilities

Cost of dinner for all employees: 60*50=3000
Cost of dinner for all guests: 60*50g
Total cost of dinner: 60*50(1+g)

Expected number of guests for each employee: (0+1+2+3+4)/5 = 2
Maximum number of guests for each employee: 4
Expected total number of guests: 2*50 = 100
Expected cost of dinner for all guests:60*100 = 6000
Expected total cost of dinner : 6000+3000 = 9000
Maximum total cost of dinner: 60*50(1+4) = 15000

Mathematical model: example2

A company is planning a corporate dinner in a park. There are 50 employees in the company. The dinner for 1 person costs $60 (including delivery cost). Each employee could invite up to 4 guests. Assume that there is 15% chance that employee brings 3 guests and 5% chance that employee brings 4 guests. Other situations (bringing 0,1,or 2 guests) occur with equal probabilities).

What is the expected cost of dinner for all? What is the maximum cost of dinner for all?

Variables: number of guests of one employee - g

g – random variable which takes values 0,1,2,3,4 with probabilities 0.25,0.25,0.25,0.15,0.05

Cost of dinner for all employees: 60*50=3000

Cost of dinner for all guests: 60*50g

Total cost of dinner: 60*50(1+g)

Expected number of guests for each employee: (0*0.25+1*0.25+2*0.25+3*0.15*4*0.05 = 1.4

Maximum number of guests for each employee: 4

Expected total number of guests: 1.4*50 = 70

Expected cost of dinner for all guests:60*70= 4200

Expected total cost of dinner : 4200+3000 = 7200

Maximum total cost of dinner: 60*50(1+4) = 15000

Morra game: find expected value of the game

two players throw out a single hand, each showing 1 to 3 fingers. If the sum of all fingers *s* is odd, then the first player wins *s* dollars. If the sum of fingers is even, then the second players wins *s* dollars.

Morra Game

Definitions:

A game is a competition among any number of persons or groups, called players, that is conducted under a prescribed set of rules with known payoffs. The rules define elementary activities or moves of the game. Each player may be allowed different moves, but each player knows the moves available to the other players.

If one player wins what another player loses, the game is called a ***zero-sum game***. A two person game is a game having only two players. Two-person, zero-sum games are called ***matrix games***.

For such a game a ***pay-off matrix*** can be created where the element (i,j) is the amount that second player pays to the first player when the second player chooses move j and the first player chooses move i.

Pay-off matrix for Morra:

		Player II		
		Move 1	*Move 2*	*Move 3*
	Move 1	2	-3	4
Player 1	*Move 2*	-3	4	-5
	Move 3	4	-5	6

Morra Game

***Pure* and *mixed* strategies.**

A *pure strategy* is a predetermined plan that prescribes the sequence of moves and countermoves that the player will make during the entire game. In a matrix game, either player has a finite set of pure strategies, although their number can be enormous. Player 1 knows player 2's move set but doesn't know for sure which element of the set player 2 has picked at the commencement of a given play of the game.

If. maximum among minimum values of the payoff matrix's columns (maximin) is equal to the minimum among maximum values of payoff matrix rows (minimax) then a pure optimal strategy exists: The optimal strategy for player 1 will be to play the row with the maximum value of the columns' minimal payoffs. The existence of a pure optimal strategy means that players can play openly. They cannot gain or lose more if their strategy is known to the other player.

If, however, maximin < minimax, then the optimal strategy can be found in *mixed strategies*. Players may gain or lose an additional payoff (at the price of additional risk) if they do not reveal their moves. The mixed strategy means that players do not know for sure which of the possible moves will take place.

Maximin < minimax => pure strategy does not exist

The optimal strategy would be found as **mixed strategy**

		Player II		
		Move 1	*Move 2*	*Move 3*
Player 1	*Move 1*	2	-3	4
	Move 2	-3	4	-5
	Move 3	4	-5	6

maxmin

minmax

Morra Game

For player 1 the goal is to maximize the payoff.

n – number of possible moves for player I
m – number of possible moves for player II
$G = \{g_{ij}\}$ **– payoff matrix,** where g_{ij} is the amount paid by player II to player I if player I uses the strategy i and player II – strategy j

A *mixed strategy* for player I can be modeled as a vector of probabilities, where the i-th element is the probability that player I chooses the move i.
$X = \{x_i\},\ i = 1,2,3 -$ **strategy of player I**

Expected gain for player I if he/she takes move1
$E(V1) = 2x_1 - \mathbf{3}x_2 + 4\,x_3$

Expected gain for player I if he/she takes move2
$E(V2) = -3x_1 + \mathbf{4}x_2 - 5\,x_3$

Expected gain for player I if he/she takes move3
$E(V3) = 4x_1 - \mathbf{5}x_2 + 6\,x_3$

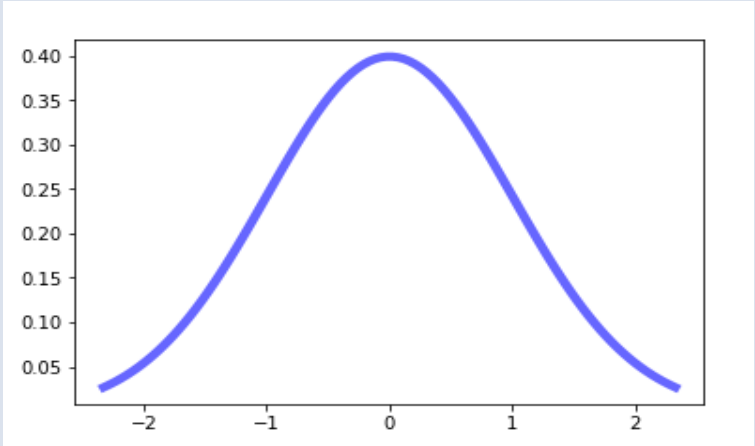
		Player II		
		Move 1	*Move 2*	*Move 3*
	Move 1	2	-3	4
Player 1	*Move 2*	-3	4	-5
	Move 3	4	-5	6

Continuous random variables

Probabilistic models of data. Why we consider probabilistic models.

One way to deal with uncertainty in mathematical model is to introduce random variables.
Mathematical variable can be deterministic (has a certain value) or stochastic(random)
(taking one of the values from a set of possible values, called a distribution of a random variable)
Distributions can be descrete and continuous.
Examples:
Normal distribution – continuous
Binomial distributio: descrete

Normal distribution is symmetrical and has thin tales.
Example of normal distribution



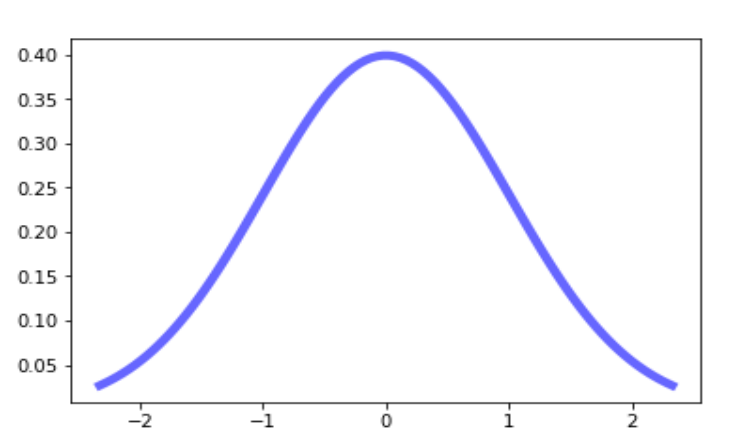
=

Probability density function: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$

Continuous random variables

Exercise: plot standard normal distribution using two methods: with statistical package and with Python alone

Normal distribution has only 2 parametrs, expected value (most probable vlue) and standard deviation (the deviation from expected value). Normal distribution is used to model a behaviour of certain quantity under high level of information uncertainty.



=

In least squares the difference between the model value and given observations (error) is modeled as random variable with normal distribution. Using the notion of random variable, we can rewrite least squares model in the following way:

$\varepsilon_i = ax_i + b - y_i$ - error term for the data point i, $\varepsilon_i \sim N(\mu, \sigma)$

$\sum_{i=0}^{n} (\varepsilon_i)^2$ - total error

$\sum_{i=0}^{n} (\varepsilon_i)^2 \rightarrow min_{\mu,\sigma}$

Discrete random variables

Simulating discrete random variables: example

Random variable X is distributed according ot the following discrete distribution:

X	1	2	3
P	0.2	0.3	0.5

Algorithm1
1.Generate random sample u from U(0,1)
2.If u<0.2 then x=1
3.If u<0.5 then x=2
4.If u<1 then x=3

Algorithm2
1.Generate random sample u from U(0,1)
2.If u<0.5 then x=3
3.If u<0.8 then x=2
4.If u<1 then x=1

Exercise: write general algorithm for sampling from a distribution given by table of possible values with probabilities

Discrete random variables

Example :rolling a die

X – result of rolling a die – random variable
Associate X with the following probability distribution

X	1	2	3	4	5	6
P	1/6	1/6	1/6	1/6	1/6	1/6

```
import numpy.random as rnd
low=1
high=7
x=rnd.randint(low, high)
```

randint(low, high) returns random integers from low value (included) to high value (excluded).

Run this code once – get 1, 2, 3, 4, 5 or 6.

i

Discrete random variables

Example :rolling a die

Run this code again and again - get 1, 2, 3, 4,5 and 6 approximately equal number of times.

```
#verify
import numpy.random as rnd
x=rnd.randint(1, 6)
cnts_dict={}
for i in range(1000):
    x=rnd.randint(1, 7)
    if x in cnts_dict:
        cnts_dict[x]=cnts_dict[x]+1
    else:
        cnts_dict[x]=1
```

Dictionary cnts_dict keeps information about how many times each value (1, 2, 3, 4, 5 or 6) has occurred:
{5: 151, 6: 163, 4: 175, 1: 171, 2: 153, 3: 187}
Next time we run it the numbers are slightly different
{2: 165, 6: 168, 3: 155, 1: 180, 4: 169, 5: 163}

Discrete random variables

Example :rolling a die

Run this code again and again - get 1, 2, 3, 4,5 and 6 approximately equal number of times.

```
#verify
import numpy.random as rnd
x=rnd.randint(1, 6)
cnts_dict={}
for i in range(1000):
    x=rnd.randint(1, 7)
    if x in cnts_dict:
        cnts_dict[x]=cnts_dict[x]+1
    else:
        cnts_dict[x]=1
```

Dictionary cnts_dict keeps information about how many times each value (1, 2, 3, 4, 5 or 6) has occurred:
{5: 151, 6: 163, 4: 175, 1: 171, 2: 153, 3: 187}
Next time we run it the numbers are slightly different
{2: 165, 6: 168, 3: 155, 1: 180, 4: 169, 5: 163}

Discrete random variables

Example :rolling a die

If for testing purposes we want to have the same outcome every time we run a test program we can initialize random number generation with the same seed:

```
import numpy.random as rnd
rnd.seed(5)
x=rnd.randint(1, 6)
cnts_dict={}
for i in range(1000):
    x=rnd.randint(1, 7)
    if x in cnts_dict:
        cnts_dict[x]=cnts_dict[x]+1
    else:
            cnts_dict[x]=1
```

We always get the same value for cnts_dict:
{4: 161, 6: 174, 1: 167, 2: 174, 5: 160, 3: 164}

i

Discrete probability distributions

Example : Playing Morra game with a computer
Suppoae that we found that the optimal strategy for the computer is the following:

X	1	2	3
P	0.2	0.3	0.5

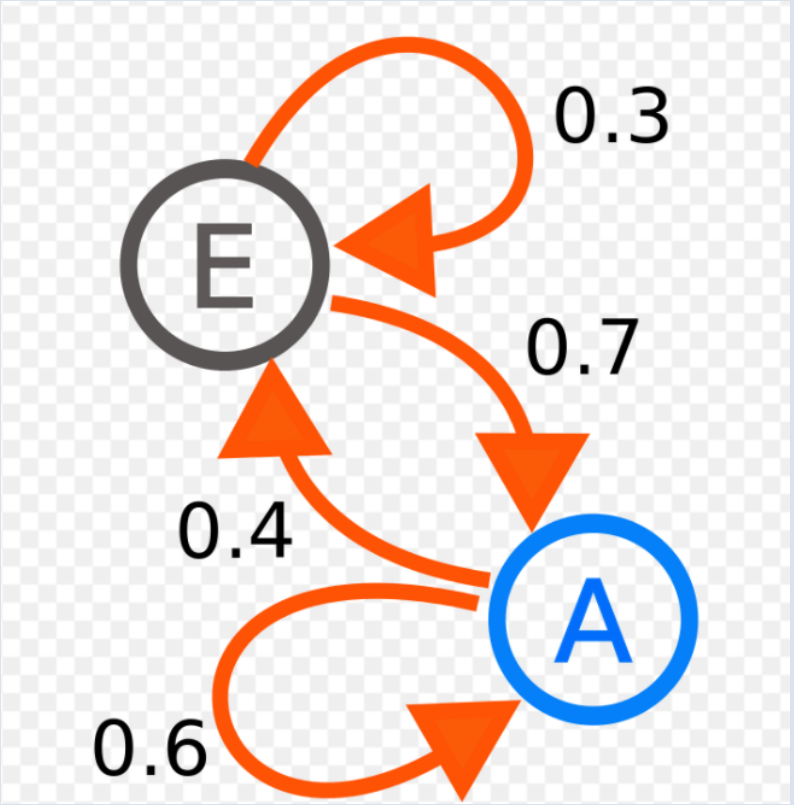
The following program executes this strategy. It generates 10 random numbers from the above probability distribution.
Python function rv_discrete from module scipy.stats returns a specified number of values generated with a discrete probability distribution

```
from scipy import stats as st
values = [1, 2, 3]
probabilities = [0.25, 0.5, 0.25]
distrib = st.rv_discrete(values=(values, probabilities))
print (distrib.rvs(size=10))
```

This is the result:
[2 2 2 2 3 3 2 2 3 2]
If we play 100 times, we get this sequence:
[2 1 2 2 3 1 1 2 3 2 1 1 1 2 2 3 2 2 1 2 3 2 2 3 2 1 2 2 3 2 2 3 2 2 3 2 3
 3 2 2 2 1 2 2 2 2 2 2 2 1 1 3 1 2 2 2 2 1 2 2 2 2 1 2 1 2 1 1 1 3 2 2 2
 2 2 2 2 3 2 3 2 2 2 2 2 1 3 2 1 2 2 3 1 2 3 2 2 3 2]

Markov chain, Markov process

stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event



Example: two-state Markov process.
states : E and A.

Each number represents the probability of the Markov process changing from one state to another state, with the direction indicated by the arrow.
For example, if the Markov process is in state A, then the probability it changes to state E is 0.4, while the probability it remains in state A is 0.6.

Markov chain, Markov process

Playing against market: two – period investment strategy

Investor wants to invest $100 for two months into bonds, stocks or cash in any proportions. Possible returns from all three types of investments are given. We assume that market conditions change every month. The matrix of transitional probabilities (probabilities that market changes its state) is given.

Market	Bonds	Stocks	MM
Low	0.0500	-0.03	0.0067
Neutral	0.04	0.003	0.067
Medium High	0.03	0.05	0.067
High	0.0250	0.082	0.067

matrix of transitional probabilities

	Low	Neutral	Medium HighHigh	
Low	**0.5**	**0.2**	**0.3**	**0.1**
Neural	**0.3**	**0.2**	**0.2**	**0.3**
Medium High	**0.1**	**0.3**	**0.2**	**0.4**
High	**0.15**	**0.2**	**0.3**	**0.35**

$$W = \{w_i\}, \qquad i = 1,2,3 - \boldsymbol{investment\ strategy}$$

w_1 - proportion of money invested in bonds, w_2 - proportion of money invested in stocks, w_3 - proportion of money invested in cash

$P = \{p_j\},\ j = 1,2,3,4 - \boldsymbol{probability\ of\ the\ market\ to\ be\ in\ certain\ state}$ **at the end of first month**

p_1 - probability of Low, p_2 -probability of Neutral, p_3 - probability of Medium High, p_4 - probability of High

$R = \{r_{ij}\}$ **– return on security *i* under market state *j***

$Q = \{q_{jk}\}$ **– probability of the market to change state j to state k**

Markov chain, Markov process

Playing against market: two – period investment strategy

Market	Bonds	Stocks	MM
Low	0.0500	-0.03	0.0067
Neutral	0.04	0.003	0.067
Medium High	0.03	0.05	0.067
High	0.0250	0.082	0.067

matrix of transitional probabilities

	Low	Neutral	Medium High	High
Low	**0.5**	**0.2**	**0.3**	**0.1**
Neural	**0.3**	**0.2**	**0.2**	**0.3**
Medium High	**0.1**	**0.3**	**0.2**	**0.4**
High	**0.15**	**0.2**	**0.3**	**0.35**

Expected return on bonds at the end of first month $E(R_b) = \sum_j r_{1j}\, p_j\, w_1$

Expected return on stocks at the end of first month $E(R_s) = \sum_j r_{2j}\, p_j\, w_2$

Expected return on cash at the end of first month $E(R_c) = \sum_j r_{3j}\, p_j\, w_3$

Total expected return at the end of first month $E(R) = E(R_b) + E(R_s) + E(R_c) = \sum_i \sum_j r_{ij}\, w_i p_j = \sum_j (p_j \sum_i r_{ij} w_i\ )$

Expected return of bonds at the end of second month $E(R_b) = \sum_k \sum_j r_{1j} w_1\ p_j q_{jk}$

…….

Total expected return at the end of second month $E(R) = E(R_b) + E(R_s) + E(R_c) = \sum_i \sum_k \sum_j r_{ij}\, w_i p_j q_{jk}$

Probabilistic modeling. Monte Carlo methods

Example : Compute area of circle using Monte Carlo method

Monte Carlo methods are numerical methods that are based on random numbers. Mathematical justification for such methods relies on two theorems, the Central Limit Theorem (CLT) and the Law of Large Numbers. Before delving into the mathematics, let us consider an intuitive example.

Suppose we want to estimate the area of a figure of an uncertain shape.

We can make a square circumscribed around this figure and compute its area. This is easy – we only need to know the length of a side of the square. At the very least, we now have an upper bound for the area of our figure. However, we can improve the estimate of the figure's area. If we randomly throw points into the square, then some points will land outside the shape and some points will land inside it. Let us say we have a mechanism for throwing the points in such a way that there is equal probability for each point to land anywhere in the square. Also suppose we have a test to determine if a point is in the shape inside the shape in the square.

If we throw enough points, the square will be roughly uniformly covered by the points

That is, eventually the number of points inside the square should be proportional to the area of the square. Then, the amount of the points inside our figure is proportional to the area of the figure.

From these two proportions we can write the equation:

N_points_sq/N_points_fig = Area_sq/Area_fig

Then, Area_fig = Area_sq * N_points_fig/ N_points_sq.

Note that we found the estimate of the area of the figure. As the number of throwing points increases, the estimate is improved.

This kind of estimation is only possible if we have a good pseudo random number generation tool.

We will use function **random** from package **random** to generate uniformly distributed pseudo random numbers between 0 and 1. We will generate both coordinates of the points separately

Mathematics of AI I: probabilistic modeling

Example : Compute area of circle using Monte Carlo method

```
from random import random
    y = random()
    x = random()
```

The preceding is equivalent to

```
    x = random(0,1)
    y = random(0,1)
```

¡Before computing the area of an unknown figure using the suggested algorithm, let us make sure that this algorithm works.

Before computing the area of an unknown figure using the suggested algorithm, let us make sure that this algorithm works. One way of ensuring it does is to consider a figure with a known area, compute this area using our method, and see if our method gives the correct value for the area. If we take a circle with radius R/2 inscribed in square with side R, we get:

$Area_{sq} = \mathrm{R}^2$

$Area_{fig} = \mathrm{pi} * (\mathrm{R}/2)^2$

pi is a constant equal to 3.1415…

The ratio $Area_{fig}/\ Area_{sq}$ should equal pi/4.

Running the simulation (the program that follows simulates the process of throwing points into a square) and see if the ratio of points landing inside the circle to the number of total points thrown into the square is equal to pi/4. This will confirm our method of estimating the area of the inscribed figure.

Mathematics of AI I: probabilistic modeling

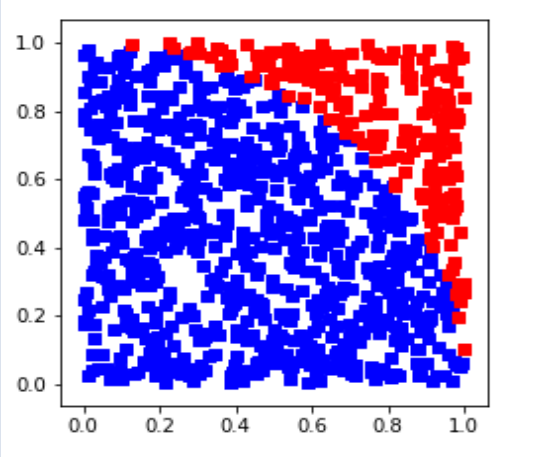
Example : Compute area of circle using Monte Carlo method

Set R = 1 and consider one fourth of the square and one fourth of the inscribed circle. Let center of the square and the circle be in the origin (coordinates of the point will be denoted x and y). Then, the condition for the point to be inside the circle is $X^2+Y^2 < 1$.

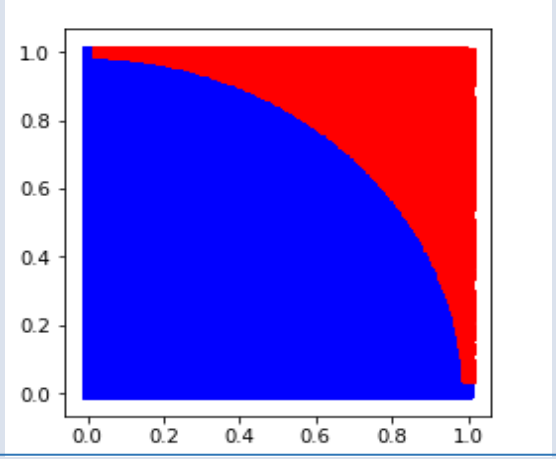
```
import random as rnd
import matplotlib.pyplot as plt

#initialization of random number generator
rnd.seed(1)
 points_in_circle = 0
n = 10**3
x_in_circle = []
y_in_circle = []
x_out_circle = []
y_out_circle = []

for _ in range(n):
    x = rnd.random()
    y = rnd.random()
    if x**2+y**2 <= 1:
        points_in_circle += 1
        x_in_circle.append(x)
        y_in_circle.append(y)
    else:
        x_out_circle.append(x)
        y_out_circle.append(y)

pi = 4*points_in_circle/n
print(pi)

fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.scatter(x_in_circle, y_in_circle, color='b', marker='s')
ax.scatter(x_out_circle, y_out_circle, color='r', marker='s')
fig.show()
```

Mathematics of AI I: probabilistic modeling

When n = 1000, we get a value of pi = 3.112.

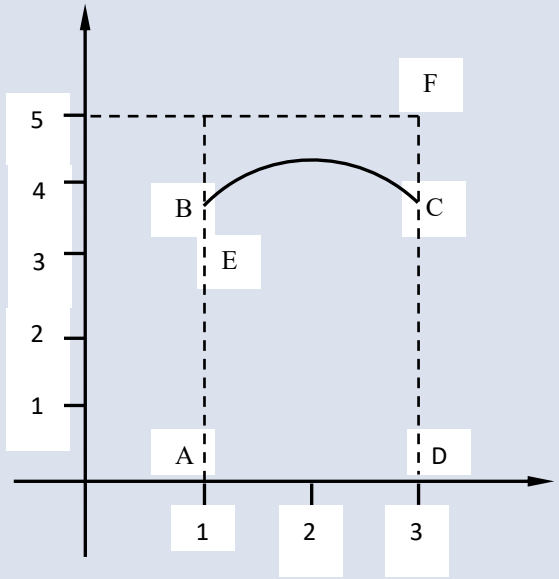


When n = 1000000, we get a value of pi =3.14138.



Mathematics of AI I: probabilistic modeling

Example: Computing area of unknown figure
BC - part of a parabola $y = -x^2 + 4x + 1$



Area S_{fig} below a certain function $f(x)$ where x is in the range [a,b] can be computed as

$$S_{fig} = \int_a^b f(x)dx$$

Mathematics of AI I: probabilistic modeling

Substituting the equation for the parabola into the above formula we get:

$$S_{ABCD} = \int_1^3 (-x^2 + 4x + 1)dx = (-\frac{x^3}{3} + 2x^2 + x) \Big|_1^3 == -\frac{3^3}{3} + 2 \times 3^2 + 3 + \frac{1}{3} - 2 - 1 =$$

$= -9 + 2 \times 9 + \frac{1}{3} - 3 = 9\frac{1}{3}$ =9.33

Another way to compute the preceding integral is by using the SymPy library,

```
import sympy as sp
x=sp.Symbol('x')
sp.init_printing(use_unicode=False,wrap_line=False)
res=sp.integrate(-x**2+4*x+1,x)
print(res)
```

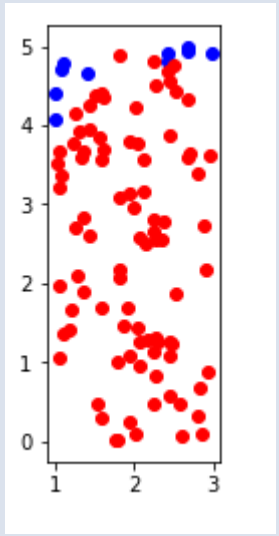
result:$-\frac{x^3}{3} + 2x^2 + x$

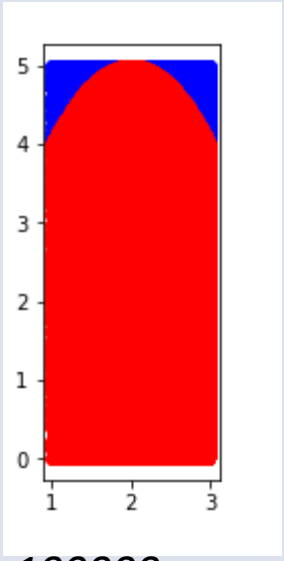
The area of parallelogram AEFD is a product of coordinates *x* and *y*:
$S_{AEFD} = 5 \times 2 = 10$.

Mathematics of AI I: probabilistic modeling

Python simulation: throwing uniformly distributed random numbers into AEFD and computing the number of points inside ABCD:

```
from random import uniform
import matplotlib.pyplot as plt
points_in = 0
n = 10**3
x_in = []
y_in = []
x_out = []
y_out = []
#x in range [1,3]
#y in range [0,5]
for _ in range(n):
    x = uniform(1,3)
    y = uniform(0,5)
    if -x**2+4*x+1 > y:
        points_in += 1
        x_in.append(x)
        y_in.append(y)
    else:
        x_out.append(x)
        y_out.append(y)
ratio=points_in/n
print(ratio)
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.scatter(x_in, y_in, color='b')
ax.scatter(x_out, y_out, color='r')
```





Estimation of the area below given curve, n=1000 and n=100000

The ratio of number of points inside the red figure to the total number of points is 0.933. This is how we compute the area of the red figure: $S_{ABCD} = S_{AEFD} * 0.933 = 10 * 0.933 = 9.33$. Compare to the analytical computation!

Definite integral of a function is the area below this function.

Mathematics of AI I: probabilistic modeling

Computing the indefinite integral is the inverse operation to computing derivatives, also called antidifferentiation. Derivatives computation in the machine learning era was discussed in the previous chapter We can imagine that the inverse operation (antidifferentiation, or computing integrals) can be performed by a computer too.

For the random variable x a so called expected value E(x) is defined. For the random variable with a discrete probability distribution, the expected value is a sum of all possible values x_i multiplied by their probabilities $p(x_i)$:

$E(x) = \sum_{i=0}^{n} p(x_i)x_i$

For the random variable with a continuous probability distribution, the expected value is the integral:

$$E(x) = \int_x p(x)xdx$$

where $p(x)$ is a probability density function

Monte Carlo methods that rely on two theorems, the **law of large numbers** and the **central limit theorem**.

The **law of large numbers** says that the average value of independent identically distributed observations of a random variable (for example, we may get the observations as a result of experiments) approaches its expected value as the number of experiments gets larger.

The **central limit theorem** says that the sum of independent, identically distributed random variables approaches a normal distribution as the number of observations gets larger.

Based on these theorems, values of integrals can be approximated, as in the area example previously. As expected values of random variables are represented by integrals the Monte Carlo methods are used to model parameter estimation if random variables are present in the model.

Mathematics of AI I: probabilistic modeling

Monte Carlo methods are approximate methods but provide good accuracy if the number of simulations is large. The growing popularity of Monte Carlo methods is caused by the increase in computer speed and the capability to produce good quality random numbers. A good quality random number generator can be considered a most important technical factor that underlies the method.

Monte Carlo methods have numerous industry applications, from analysis of social and natural processes to pricing of financial instruments and estimation of financial portfolios risk and returns.

Mathematics of AI I: Bayes Rule

Proposition - a statement that expresses a concept that can be true or false
P(A) – ***unconditional, or prior probability*** that proposition A is true
As soon as some new information B is known, we have to reason with the ***conditional or posterior probability*** of A given B, P(A|B) instead of P(A). Important: we say P(*A*|*B*) if only information we know is B. As soon as we know C we have to write P(A|B and C)

Example:
Cavity denotes the proposition that a particular patient has a cavity
P(*Cavity*) = 0.1 means that in the absence of any other information a probability of 10% (is assigned to the event *Cavity.*
Toothache - denotes the proposition that a particular patient has a toothache
P(*Cavity*|*Toothache*) =0.8 probability that patient has cavity given that this patient has Toothache is 80%.

Product Rule:
P(A and B) is **joint probability** of A and B
P(A and B) = P(A|B)*P(B)= P(B|A)*P(A) (*)
Or in the form P(A|B)= P(A|B)/ P(A|B), it holds for P(B)>0

Axioms of probability:
1. 0<=P(A)<=1
2.P(True)=1, P(False)=0
3.P(A or B)=P(A)+P(B)-P(A and B)

Bayes Rule:
Equating both sides of (*) we get: P(A|B)=P(B|A)*P(A) / P(B)

Mathematics of AI I: MLE and MAP

MLE – maximum Likelihood Estimation

MAP-Maximum A Posteriori Estimation

Calibration (or fitting) of the model means that parameters of the should be estimated in such a way that the model fits given data.

MLE: find such parameters of the model that the probability of the given data sample is maximized.

Let Z be a random variable with independent observations $z_1, z_2, \dots, z_n$ The distribution of Z is not known, but it belongs to a family of distributions $f(z_1, ..z_n|\beta)$ with some unknown parameter β

$f(Z, \beta)$ - joint density function for Z and β.

β is a parameter that has to be estimated through observations of Z. β statistically depends on Z.

MLE :maximization of probability of observation set Z to occur ($f(Z|\beta)$).

MAP :maximization of posterior probability $f(\beta|Z)$

Bayes rule connects likelihood function with posterior probability: $f(\beta|Z) = \frac{f(Z|\beta)f(\beta)}{f(Z)}$

$f(Z)$ can be skipped in maximization because it does not depend on β.

$f(\beta)$ – prior distribution of parameter β.

If we do not have any knowledge about $f(\beta)$ we assume uniform distribution (which has constant density) , and maximizing $f(\beta|Z)$ is equivalent to maximizing likelihood function $f(Z|\beta)$.

If $f(\beta)$ is not constant than MAP and MLE will result in different estimate of β.

MLE is a particular case of MAP when prior is uniform!

If we knew the distribution function $f(\beta|Z)$ then we could compute estimation $\hat{\beta}$ easily, but $f(\beta|Z)$ is not available and so we have to estimate it through observations $z_1, z_2, \dots, z_n$. As a result we get a point estimate that is highly sensitive to the quality of $z_1, z_2, \dots, z_n$ observations (In other words, both MLE and MAP give point estimate)

Mathematics of AI I: MLE

How do we compute $f(Z|\beta)$?. Because observations of Z are iid their joint probability distribution can be written as a product of probability distributions of each observation:

$$f(z_1, z_n|\beta) = \prod_i f(z^i|\beta)$$

Monotonic transformation of the above expression will not change the solution of maximization problem, so we apply logarithmic transformation:

$$\log\left[\prod_{i=1}^{n} f(z^i|\beta)\right] = \sum_{i=1}^{n} \log f(z^i|\beta)$$

What is the probability of a single observation in the case of airplane recognition problem? It is a binary event and can be modeled as Bernoulli distribution $(1 \text{ with probability } \mathrm{p}, 0 \text{ with probability } (1-\mathrm{p})$.
If we have *n* observations of a random event than this situation can be modeled as a Binomial distribution. Substituting the probability of a binomial distribution into the preceding formula (MLE estimator) we get the following:

$$L = \prod_{i=1}^{N} p^{y_i} (1-p)^{1-y_i}$$

$$logL = \sum_{i=1}^{N} y_i \log p + \sum_{i=1}^{N} (1-y_i) \log (1-p)$$

This is typical loss function in binary classification methods!

In linear regression the error term (difference between model and data) is assumed to be normally distributed. MLE estimator will result in sum od squares minimization (**Exercise**: prove it!)

Mathematics of AI I: Binomial Distribution

*In probability theory and statistics, the **binomial distribution** with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent experiments, each asking a yes–no question, and each with its own boolean-valued outcome: success/yes/true/one (with probability p) or failure/no/false/zero (with probability q = 1 − p). A single success/failure experiment is also called a Bernoulli trial or Bernoulli experiment and a sequence of outcomes is called a Bernoulli process;*

In general, if the random variable X follows the binomial distribution with parameters $n \in \mathbb{N}$ and $p \in [0,1]$, the probability of getting exactly k successes in n independent Bernoulli trials is given by the probability mass function:

$$p(k,n,p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Where $\binom{n}{k} = \frac{n!(n-k)!}{k!}$

Mathematics of AI I: Naïve Bayes

Naive Bayes methods - set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of *conditional independence* between every pair of features given the class label.

Given class label y and dependent feature vector Z= (z_1, z_n):

$$\mathrm{P}(y|z_1,..z_n) = \frac{\mathrm{P}(y)\prod_i \mathrm{P}(z_i|\mathrm{y})}{\mathrm{P}(z_1,..z_n)}$$

Note that because $z_1,..z_n$ is given the probability $\mathrm{P}(z_1,..z_n)$ is constant.

MAP:

$$argmax_{\mathrm{y}}\ \mathrm{P}(y|z_1,..z_n) = argmax_{\mathrm{y}}\ \mathrm{P}(y)\prod_i \mathrm{P}(z_i|\mathrm{y})$$

We are looking for such y that maximizes $\mathrm{P}(y)\prod_i \mathrm{P}(z_i|\mathrm{y})$.

$\mathrm{P}(y)$ = relative frequency of class y in training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution $\mathrm{P}(z_i|\mathrm{y})$

Mathematics of AI I: Naïve Bayes

Advantages:

-simple
-works well on real word applications such as document classification and spam filtering.
-require a small amount of training data to estimate the necessary parameters
-fast compared to more sophisticated

Disadvantages:

-probability outputs from predict_proba are not precise

Mathematics of AI I: Naïve Bayes Example

Classifying Iris data using Naïve Bayes under Gauusian assumption on likelihood function.

class **sklearn.naive_bayes.GaussianNB**(priors=None, var_smoothing=1e-09)

priors - Prior probabilities of the classes. If specified the priors are not adjusted according to the data.
var_smoothing - Portion of the largest variance of all features that is added to variances for calculation stability.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d" % (X_test.shape[0], (y_test != y_pred).sum()))
```

Result: 4 misclassified points out of 75 total points

Mathematics of AI I: Naïve Bayes Example (cont.)

NBGaussian Attributes:

class_count_ array, shape (n_classes,), number of training samples observed in each class.

result: array([29., 20., 26.])

class_prior_ array, shape (n_classes,), probability of each class.

result: array([0.38666667, 0.26666667, 0.34666667])

classes_ array, shape (n_classes,), class labels known to the classifier

result: array([0, 1, 2])

epsilon_ float, absolute additive value to variances, see https://dl.acm.org/doi/10.1109/WI-IAT.2013.80

sigma_ array, shape (n_classes, n_features), variance of each feature per class

result: array([[0.10321047, 0.13208086, 0.01629013, 0.00903686],
[0.256275 , 0.0829 , 0.255275 , 0.046],
[0.38869823, 0.10147929, 0.31303255, 0.04763314]])

theta_ array, shape (n_classes, n_features), mean of each feature per class

result:array([[4.97586207, 3.35862069, 1.44827586, 0.23103448],
[5.935 , 2.71 , 4.185 , 1.3],
[6.77692308, 3.09230769, 5.73461538, 2.10769231]])

Mathematics of AI I: 20 newsgroups dataset

Homework: use naïve Bayes classifier to build a classifier for identifying text data
Use toy dataset **20 newsgroups**

The 20 newsgroups dataset comprises around 18000 newsgroups posts on 20 topics split in two subsets: one for training (or development) and the other one for testing (or for performance evaluation). The split between the train and test set is based upon a messages posted before and after a specific date.

https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

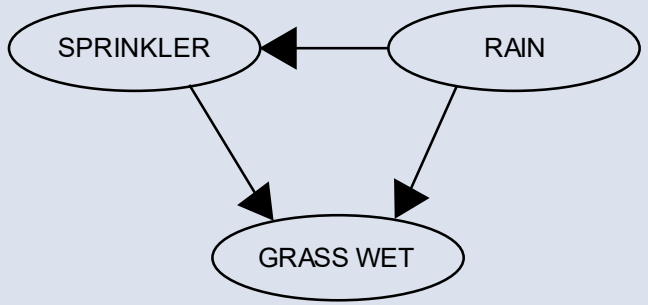
Mathematics of AI I: Belief Networks

A **Bayesian network**, **Bayes network**, **belief network**, **decision network**, **Bayes(ian) model** or **probabilistic directed acyclic graphical model** is a probabilistic graphical model that represents a set of variables and their conditional dependecies using Directed Acyclic Graphs (DAGs).

Example from Wikipedia:

Two events can cause grass to be wet: an active sprinkler or rain. Rain has a direct effect on the use of the sprinkler (when it rain the sprinkler usually is not active). This situation can be modeled with a Belief Network. Each variable has two possible values, T (for true) and F (for false). The model can answer questions about the presence of a cause given the presence of an effect (so-called inverse probability) like "What is the probability that it is raining, given the grass is wet?"

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99



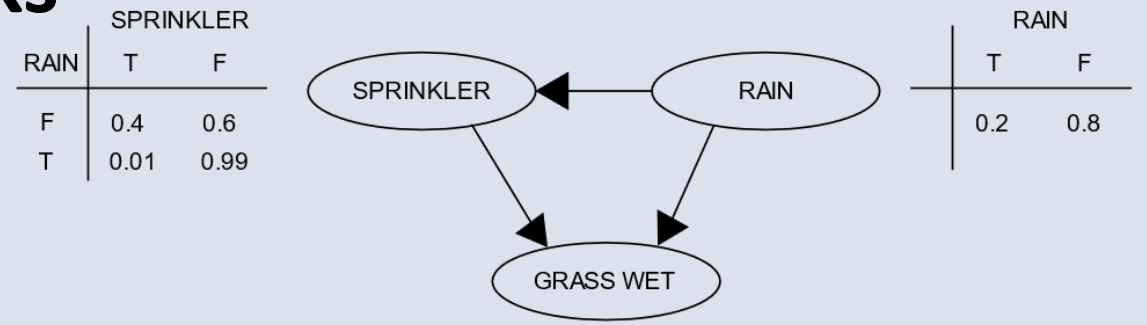
	RAIN	
T	F	
0.2	0.8	

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Mathematics of AI I: Belief Networks

Let us use the following event:

G = “Grass Wet”
S=”Sprinkler turned on”
R=”Raining”



The joint probability function: P(G,S,R)=P(G|S,R)P(S|R)P(R)

GRASS WET

SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Conditional probability of rain given the grass is wet:

$$P(R=T|G=T)=\frac{P(G=T,R=T)}{P(G=T)}=\frac{\sum_{S=\{T,F\}}P(G=T,S,R=T)}{\sum_{R,S=\{T,F\}}P(G=T,S,R)}$$

$$=\frac{P(G=T,S=T,R=T)+P(G=T,S=F,R=T)}{P(G=T,S=T,R=T)+P(G=T,S=T,R=F)+P(G=T,S=F,R=T)+P(G=T,S=F,R=F)}$$

Using the expansion for the joint probability function P(G,S,R) and the conditional probabilities from the tables in the diagram one can evaluate each term in the sums in the numerator and denominator. For example,

$$P(G=T,S=T,R=T)=P(G=T|S=T,R=T)P(S=T|R=T)P(R=T)=0.99*0.01*0.2=0.00198$$

$$P(G=T,S=F,R=T)=P(G=T|S=F,R=T)P(S=F|R=T)P(R=T)=0.8*0.99*0.2=0.1584$$

then the numerical results (subscripted by the associated variable values) are $P(R=T|G=T)=$

$$\frac{0.00198_{TTT}+0.1584_{TFT}}{0.00198_{TTT}+0.288_{TTF}+0.1584_{TFT}+0.0_{FFF}}=\frac{891}{2491}=35.77..\%$$

Mathematics of AI I: Belief Networks in scikit learn

Restricted Boltzmann machines
https://scikit-learn.org/stable/modules/neural_networks_unsupervised.html#rbm
papers
"A fast learning algorithm for deep belief nets" G. Hinton, S. Osindero, Y.-W. Teh, 2006
"Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient" T. Tieleman, 2008