2019 Introduction to Machine Learning Program Assignment #1

# Naïve Bayes

~~Machine Learning? Nah, probability!~~

TA: Evan Chang, toosyou.tw@gmail.com

# Goals

1. Naïve Bayes

2. Basic workflow of ML

# Goals

1. Naïve Bayes

2. Basic workflow of ML

# Goals

1. Naïve Bayes

2. Basic workflow of ML

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Data Input – 1/2

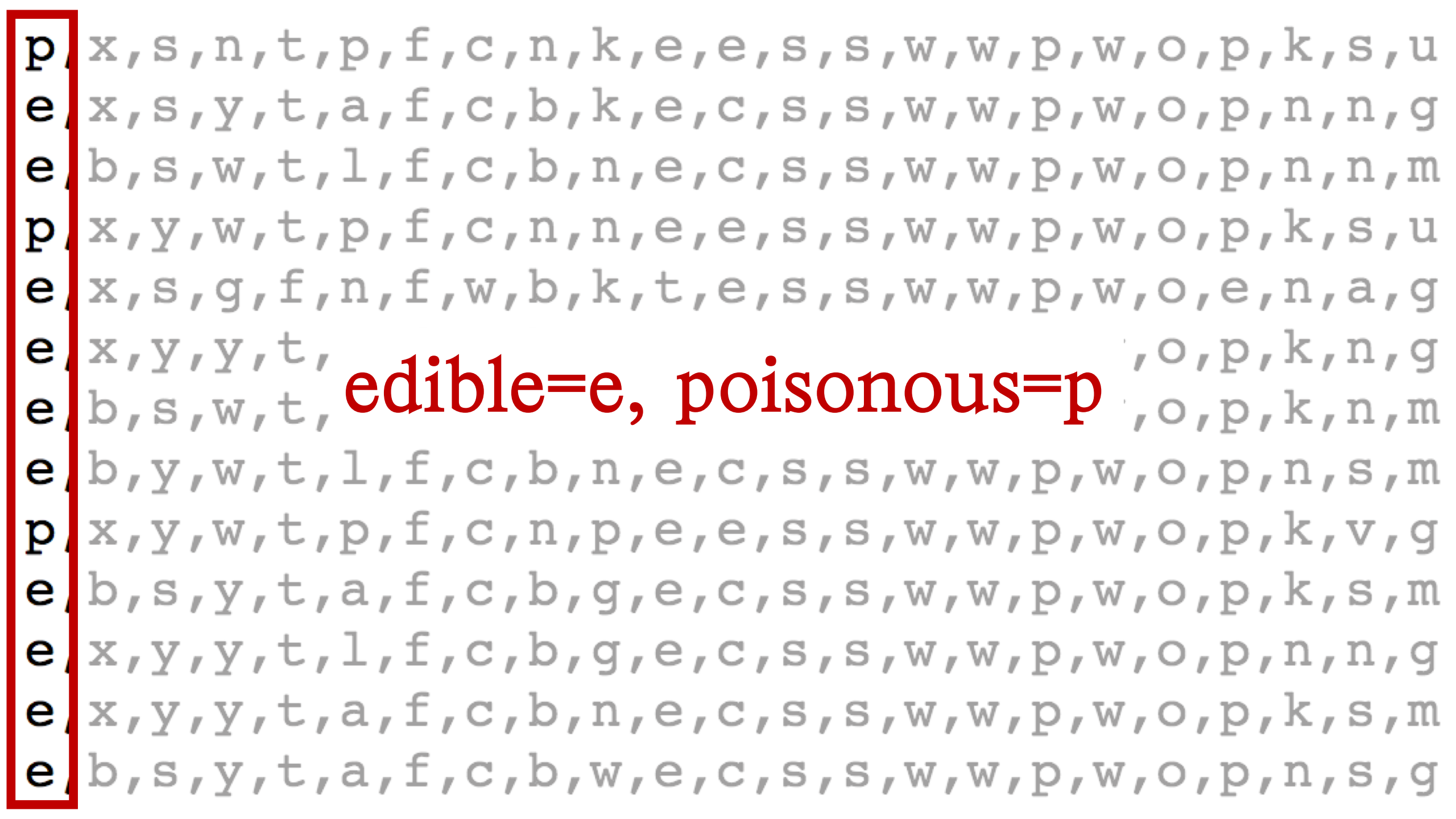
## Mushroom Data Set

*Download*: Data Folder, Data Set Description

**Abstract**: From Audobon Society Field Guide; mushrooms described in terms of physical characteristics; classification: poisonous or edible

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 8124 | **Area:** | Life |
| **Attribute Characteristics:** | Categorical | **Number of Attributes:** | 22 | **Date Donated** | 1987-04-27 |
| **Associated Tasks:** | Classification | **Missing Values?** | Yes | **Number of Web Hits:** | 469094 |

p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u
e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g
e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m
p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u
e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g
e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g
e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m
e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m
p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g
e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m
e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g
e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,m
e,b,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g


p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u
e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g
e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m
p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u
e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g
e,x,y,y,t,,,o,p,k,n,g
e,b,s,w,t,,,o,p,k,n,m
e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m
p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g
e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m
e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g
e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,m
e,b,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g
edible=e, poisonous=p

- Attribute Information
    1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
    2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
    3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,
       pink=p,purple=u,red=e,white=w,yellow=y
    4. bruises?: bruises=t,no=f
    5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,
       musty=m,none=n,pungent=p,spicy=s
    6. gill-attachment: attached=a,descending=d,free=f,notched=n
    7. gill-spacing: close=c,crowded=w,distant=d
    8. gill-size: broad=b,narrow=n
    9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,
       green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
    10. stalk-shape: enlarging=e,tapering=t
    11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r, **missing=?**
    12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
    13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
    14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,
        pink=p,red=e,white=w,yellow=y
    15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,
        pink=p,red=e,white=w,yellow=y
    16. veil-type: partial=p,universal=u
    17. veil-color: brown=n,orange=o,white=w,yellow=y
    18. ring-number: none=n,one=o,two=t
    19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,
        none=n,pendant=p,sheathing=s,zone=z
    20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,
        orange=o,purple=u,white=w,yellow=y
    21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
    22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

# Data Input – 2/2

## Iris Data Set

*Download*: Data Folder, Data Set Description

**Abstract**: Famous database; from Fisher, 1936

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 150 | **Area:** | Life |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 4 | **Date Donated** | 1988-07-01 |
| **Associated Tasks:** | Classification | **Missing Values?** | No | **Number of Web Hits:** | 2870168 |

- Attribute Information
    1. sepal length in cm
    2. sepal width in cm
    3. petal length in cm
    4. petal width in cm
    5. class:
        - Iris Setosa
        - Iris Versicolour
        - Iris Virginica

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Data Visualization – 1/2

Show the data distribution by

- Average
- Standard Deviation
- Value Frequency

of every feature.

# Data Visualization – 2/2

Split data based on their labels (targets)

And show the data distribution of each feature again.

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Data Preprocessing

- Drop features with any missing value.
- Transform data format and shape
- Shuffle the data.

# Data Preprocessing – Bonus

- Any other transformation that helps

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Model Construction

Construct two Naïve Bayes classifiers

WITHOUT any package-provided model.

# Model Construction

$$M(\mathbf{q}) = \underset{Y \in \mathbb{T}}{\operatorname{argmax}} \left[ \log P(Y) + \sum_{i=1}^{m} \log P\left(X_i | Y\right) \right]$$

# Model Construction

For categorical data,

compare results with and without Laplace smoothing.

# Model Construction

For numerical data,

assume $P(X_i|Y)$ follows 1D Gaussian distribution.

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Train-Test-Split

1. Holdout Validation,
   $|TrainingSet|:|TestingSet| = 7:3$

2. K-fold Cross-validation,
   $K = 3$

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Results

1. Confusion Matrix
2. Accuracy
3. Sensitivity (Recall)
4. Precision

# Objective

1. Data Input
2. Data Visualization
3. Data Preprocessing
4. Model Construction
5. Train-Test-Split
6. Results
7. Comparison & Conclusion

# Submission & Scoring Policy

A .ZIP to newE3 with

1. Report in PDF or HTML
    - All the things mentioned
    - Explanation
    - Name, Student ID
2. Source Codes
    - C/C++, python2/python3, MATLAB, JAVA

NO ML packages allowed

However,

numerical and visualization packages are allowed

Take python3 for example:

- sklearn
- numpy
- pandas
- matplotlib

EXCEL is fine for visualization, too.

TA will ask you to demo,
if there is any problem with your code.

Otherwise, no demo.

Plagiarizing (抄襲, aka copy-paste) is **NOT** allowed, of course.

作業零分

- First time caught, ZERO on that HW.

被當掉

- Second time, you will **FAIL** this class.

# Score Normalization

- Mean -> 80
- Standard Deviation -> 15

# Some tools for python

File Edit View Run Kernel Tabs Settings Help

Files Running Commands Cell Tools Tabs

> notebooks

| Name | Last Modified |
|---|---|
| Data.ipynb | an hour ago |
| Fasta.ipynb | a day ago |
| Julia.ipynb | a day ago |
| Lorenz.ipynb | seconds ago |
| R.ipynb | a day ago |
| iris.csv | a day ago |
| lightning.json | 9 days ago |
| lorenz.py | 3 minutes ago |

Lorenz.ipynb | Terminal 1 | Console 1 | Data.ipynb | README.md

Code — Python 3

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

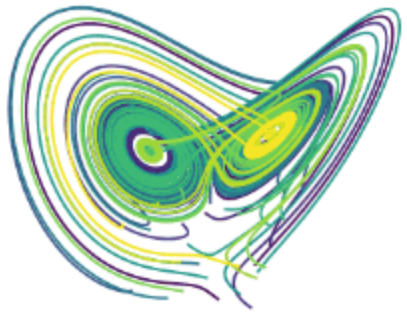
In [4]:
```
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

Output View

sigma 10.00

beta 2.67

rho 28.00



lorenz.py

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

File Edit View Run Kernel Tabs Settings Help

Files Running Commands Cell Tools Tabs

> notebooks

| Name | Last Modified |
|---|---|
| Data.ipynb | an hour ago |
| Fasta.ipynb | a day ago |
| Julia.ipynb | a day ago |
| Lorenz.ipynb | seconds ago |
| R.ipynb | a day ago |
| iris.csv | a day ago |
| lightning.json | 9 days ago |
| lorenz.py | 3 minutes ago |

Lorenz.ipynb | Terminal 1 | Console 1 | Data.ipynb | README.md

Code — Python 3

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

In [4]:
```
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

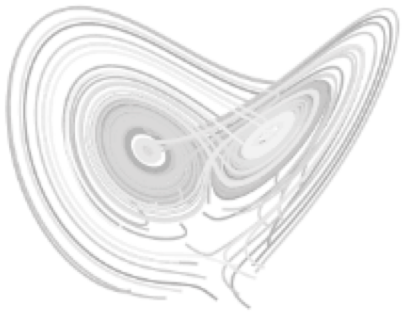
# JuypterLab

Output View

sigma 10.00

beta 2.67

rho 28.00



lorenz.py

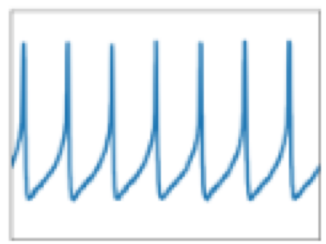
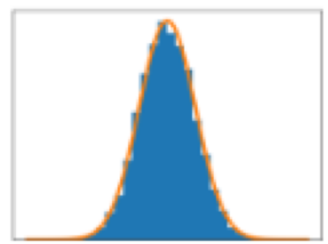
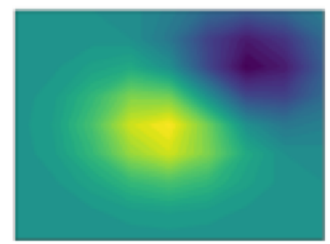
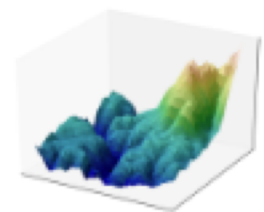
```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

# matplotlib

Version 3.1.1



Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.