Social Data Science: Machine Learning & Econometrics

Exercise class 2

February 21, 2020

Todays quick warmup

Q: Write a *generator* `pascal()` that *yields* subsequent rows in Pascals Triangle.

row 1				1				
row 2			1		1			
row 3		1		2		1		
row 4	1		3		3		1	
⋮				⋮				

Your generator should take 0 arguments, `next()` should give you the next row in the triangle.

Bonus: Sierpiński's triangle can be drawn by plotting only the odd numbers of pascals triangle as black dots. Do this for $n = 1024$[1] rows of pascals triangle.

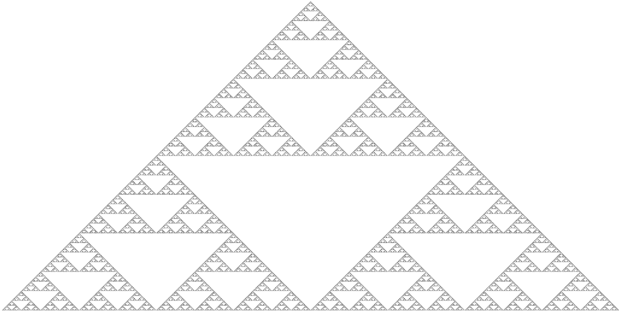
[1] Some of you might need to think about the maximum size of integers to get n above 50-60

Todays quick warmup - solution

Yield the first row [1] manually, then forever pad the last row with 0's and yield the next row.

```
def pascals():
    row = [1]
    yield row
    while True:
        row = [0] + row + [0]
        row = [i+j for i,j in zip(row[:-1], row[1:])]
        yield row
```

Todays quick warmup - bonus solution



Todays quick warmup - bonus solution

Make a big 0-matrix and fill in as we go.

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

def plot_sierpinski(n):
    M = np.zeros(shape = (n, 2*n))
    midpoint = int(np.ceil(2*n / 2))
    triangle = pascals()

    for row in range(n):
        elems = np.array([t%2!=0 for t in next(triangle)])
        insert = np.insert(elems, range(1,row+1) ,0)
        M[row, midpoint-row:midpoint+row+1] = insert
    return M

M = plot_sierpinski(1000)
plt.imshow(M)
```

Last lecture in a nutshell

Once again lots of stuff was covered in the lecture

- **Dimensionality reduction:** PCA, LDA, t-SNE and UMAP
- **Clustering:** K-means, C-means, Mixture models, ..., DBSCAN

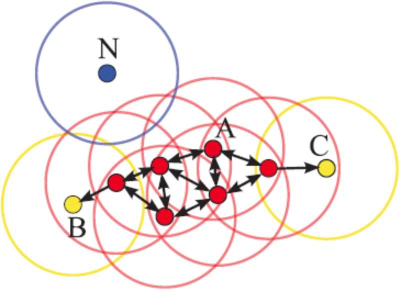
- **PCA:** Reduce dimensionality by projecting on "variance explaining" basis.
- Caveats: finds best *global linear* projection.
- **LDA:** Similar to PCA, but use information on class labels to target *maximum class separation*.

```
# Import Pipeline, PCA
# and StandardScaler
k = 10
model = Pipeline([
    ('scale', StandardScaler()),
    ('pca', PCA(n_components=k))
])

model.fit_transform(X)
```

Last lecture in a nutshell

DBSCAN: independently sets the number of clusters, fits weird-shaped clusters and allows for noise observations.



- *Caveats:* non-deterministic, uses (by default) Euclidian distance, so suffers from curse of dimensionality, has a fixed ε, so clusters of different density cause problems.
- Note also that the combination of Euclidian distances and fixed ε requires common scale to be meaningful.