Homework 4

1.

To show that $||x||_A = \sqrt{x^T A x}$ is a norm:

(i) by property of A that A is positive definite $x^T A x \geq 0$, then $||x||_A$ is nonnegative.

(ii) also with property of A, $x^T A x \geq 0$ for all $x \in R^d$ and with equality with if and only if $x = 0$, $||x||_A = 0$ is if and only if $x = 0$.

(iii) homogeneous:

For any $t \in R$,

$$||tx||_A = \sqrt{(tx)^T A (tx)} = \sqrt{t^2 x^T A x} = |t|\sqrt{x^T A x} = |\mathrm{t}|\,||x||_A$$

(iv) the triangle inequality:

Firstly, we can show that since $||x||_A = \sqrt{x^T A x}$, then

$$||x||_A = \sqrt{x^T A x} = \sqrt{x^T U U^T x} = ||U^T x||_2$$

Hence, the prof of triangle inequality can be transferred to

$$||x+y||_A \leq ||x||_A + ||y||_A$$

$$\Leftrightarrow ||U^T x + U^T y||_2 \leq ||U^T x||_2 + ||U^T y||_2$$

$$\Leftrightarrow ||\alpha + \beta||_2 \leq \underbrace{||\alpha||_2}_{\alpha = U^T x} + \underbrace{||\beta||_2}_{\beta = U^T y}$$

And here is to show ℓ_2 norm triangle inequality:

$$||\alpha + \beta||_2 = \sqrt{\sum_{i=1}^{d} (\alpha_i + \beta_i)^2}$$

$$= \sqrt{\sum_{i=1}^{d} ({\alpha_i}^2 + {\beta_i}^2 + 2\alpha_i \beta_i)}$$

$$\leq \underbrace{\sqrt{\sum_{i=1}^{d}({\alpha_i}^2+{\beta_i}^2)+2\sqrt{\sum_{i=1}^{d}{\alpha_i}^2\sum_{i=1}^{d}{\beta_i}^2}}}_{using\ Cauchy\ Inequality}$$

$$=\sqrt{\left(\sqrt{\sum_{i=1}^{d}{\alpha_i}^2}+\sqrt{\sum_{i=1}^{d}{\beta_i}^2}\right)^2}$$

$$=\sqrt{\sum_{i=1}^{d}{\alpha_i}^2}+\sqrt{\sum_{i=1}^{d}{\beta_i}^2}$$

$$=\left|\left|\alpha\right|\right|_2+\left|\left|\beta\right|\right|_2$$

As for $Cauchy\ Inequality$ part, here is detail

$$2\sum_{i=1}^{d}(\alpha_i\beta_i)\leq 2\sqrt{\sum_{i=1}^{d}{\alpha_i}^2\sum_{i=1}^{d}{\beta_i}^2}$$

So as shown above it satisfies triangle inequality.

Based on 4 properties, the $\left|\left|x\right|\right|_A=\sqrt{x^TAx}$ is a norm.

And based on the HW1, we can know how it can yield to distance.

2.

(a)

With

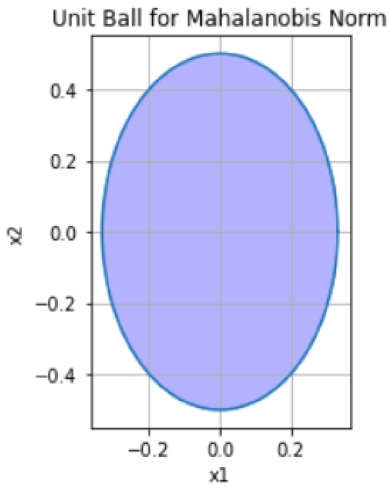
$$\mathrm{A}=\begin{pmatrix}9 & 0\\ 0 & 4\end{pmatrix}$$

Then

$$\left|\left|x\right|\right|_A=\sqrt{x^TAx}=\sqrt{9x_1^2+4x_2^2}\leq 1$$

$$\Longrightarrow 9x_1^2 + 4x_2^2 \leq 1$$

That is an ellipse and the figure drawn by Python is shown



And the blue area is for unit ball associated with Mahalanobis norm

$$K = \{x \in R^d : ||x|| \leq 1\}$$

(b)

Similar to what we've done in HW1, using properties of norm, we can obtain

$$||\theta x + (1-\theta)y|| \leq ||\theta x|| + ||(1-\theta)y|| \text{ (triangle inequality)}$$

$$= |\theta|||x|| + |(1-\theta)|||y|| \text{ (homogeneous)}$$

And since $x, y \in K$, we have $||x|| \leq 1, ||y|| \leq 1$

Hence,

$$|\theta|||x|| + |(1-\theta)|||y|| = \theta||x|| + (1-\theta)||y||$$

$$\leq \theta + (1-\theta) = 1$$

$$\Rightarrow ||\theta x + (1-\theta)y|| \leq 1$$

$$\Rightarrow ||\theta x + (1-\theta)y|| \in K, for\ any\ 0 < \theta < 1$$

In other word, the unit ball with associated norm is a convex set

(c)

The conclusion is $K' \subseteq K$, prof is as follows

For any $x \in K'$, then we have $||x||' \leq 1$

Then,

$$||x|| \leq ||x||' \leq 1$$

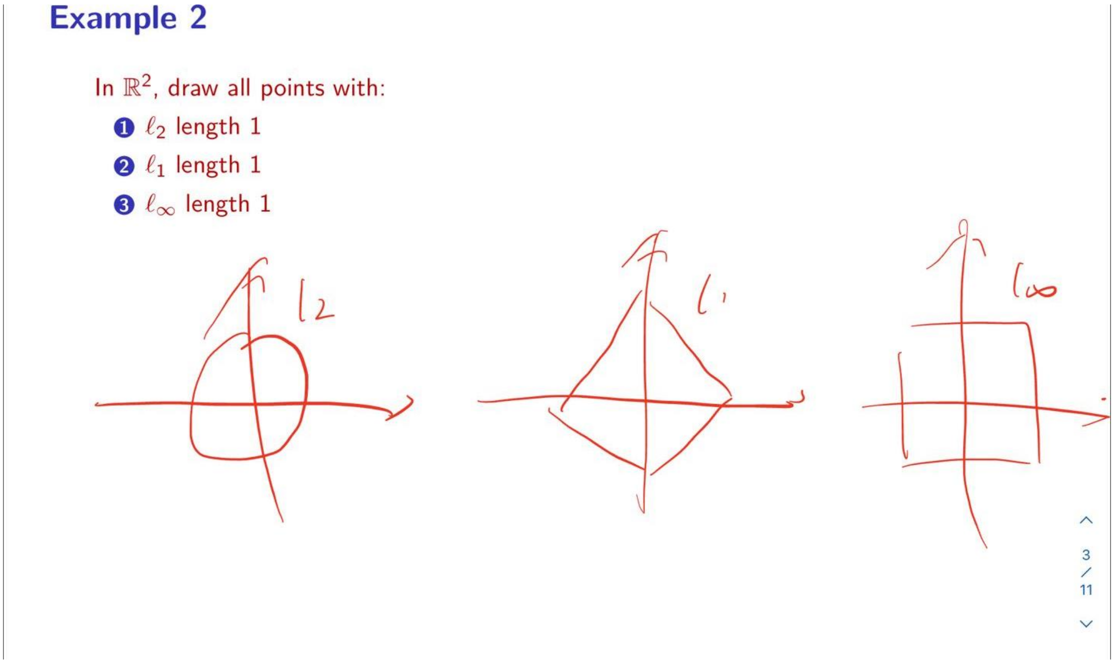
$$\Rightarrow x \in K$$

Therefore $K' \subseteq K$

(d)

It should be the unit ball with ℓ_1 norm.

Similar to the lecture,



this is a sketch when dimension is only 2 for simplicity. Obviously, for any dimension, it's still working, that ℓ_1 norm unit ball contains the least area.

3.

By definition, we can easily know that

$$\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q_i = 1, for\ every\ p\epsilon S$$

We use Lagrange formula to calculate,

$$\mathcal{L} = \sum_{p\epsilon S} \sum_{i=1}^{m} p_i ln \frac{p_i}{q_i} + \lambda \left(\sum_{i=1}^{m} q_i - 1 \right)$$

Therefore, we calculate derivatives and let them be 0

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial q_i} = \sum_{p\epsilon S} \dfrac{-p_i}{q_i} + \lambda = 0, \quad for\ every\ 1 \le i \le m \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = \sum_{i=1}^{m} q_i - 1 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} \dfrac{\sum_{p\epsilon S} p_i}{q_i} = \lambda, \quad for\ every\ 1 \le i \le m \\ \sum_{i=1}^{m} q_i = 1 \end{cases}$$

$$\Rightarrow \begin{cases} \dfrac{\sum_{p\epsilon S} p_i}{\lambda} = q_i, \quad for\ every\ 1 \le i \le m \\ \sum_{i=1}^{m} q_i = 1 \end{cases}$$

Then we combine these two and using the property mentioned at the beginning,

$$\sum_{i=1}^{m} q_i = \sum_{i=1}^{m} \frac{\sum_{p\epsilon S} p_i}{\lambda} = \frac{\sum_{p\epsilon S} \sum_{i=1}^{m} p_i}{\lambda}$$

$$\Rightarrow 1 = \frac{\sum_{p\epsilon S} 1}{\lambda} = \frac{|S|}{\lambda}$$

$$\Rightarrow \lambda = |S|$$

Then

$$q_i = \frac{\sum_{p\epsilon S} p_i}{\lambda} = \frac{\sum_{p\epsilon S} p_i}{|S|}, for\ every\ 1 \leq i \leq m$$

Thus, we can obtain conclusion the desired q is

$$q = (q_1, q_2, \dots, q_m) = (\frac{\sum_{p\epsilon S} p_1}{|S|}, \frac{\sum_{p\epsilon S} p_2}{|S|}, \dots, \frac{\sum_{p\epsilon S} p_m}{|S|})$$

4.

given the hint using Jensen's inequality, and using $f(z) = -lnz$, we have

$$K(p,q)$$

$$= E_{X\sim p}\left(\frac{lnp(x)}{\ln q(x)}\right)$$

$$= E_{X\sim p} - \left(\frac{lnq(x)}{\ln p(x)}\right)$$

$$\geq -\ln\left(E_{X\sim p}\frac{q(x)}{p(x)}\right)$$

$$= -\ln\left(\sum_{x\in X} p(x)\frac{q(x)}{p(x)}\right)$$

$$= -\ln\left(\sum_{x\in X} q(x)\right) = 0$$

Therefore, we showed the nonnegativity.

$$K(p,q) \geq 0$$

5.

(a)

Using data in the table and the maximum likelihood estimation

$$\lambda = \frac{\sum_k N_k k}{\sum_k N_k} = \frac{0 \cdot 22 + 1 \cdot 66 + \cdots + 8 \cdot 10}{500} = 3.154$$

```
a=[22, 66, 106, 115, 85, 55, 28, 13, 10]
b=[i*a[i] for i in range(len(a))]
sum(b)
```

```
1577
```

```
sum(b)/500
```

```
3.154
```

(b)

Using the formula, $Pr(\mathrm{X} = \mathrm{k}) = \frac{e^{-\lambda}\lambda^k}{k!}$, and Python

```
import numpy as np
import math
k=[round(500*np.exp(-3.154)*(3.154**i)/math.factorial(i),2) for i in range(len(a))]
k
```

```
[21.34, 67.31, 106.14, 111.59, 87.99, 55.51, 29.18, 13.15, 5.18]
```

We can have

k	0	1	2	3	4	5	6	7	8	≥9
N_k	21.34	67.31	106.14	111.59	87.99	55.51	29.18	13.15	5.18	2.61

And they are pretty close to original hence prove we' done right.

6.

(a)

Since $X \sim U_\lambda(\lambda)$, then pdf is

$$f_X(x) = \begin{cases} \frac{1}{\lambda}, 0 \leq x \leq \lambda \\ 0, others \end{cases}$$

And the cdf is

$$F_X(x) = \begin{cases} 0, x < 0 \\ \dfrac{x}{\lambda}, 0 \le x \le \lambda \\ 1, x > 1 \end{cases}$$

(b)

Using maximum likelihood estimation,

$$L(\lambda) = \prod_{i=1}^{n} f_{X_i}(x) = \begin{cases} \dfrac{1}{\lambda^n}, 0 \le x_1, x_2, \dots, x_n \le \lambda \\ 0, others \end{cases}$$

For this case, unlike usual way to compute derivative, we found that this function increased as λ decreased with $0 \le x_1, x_2, \dots, x_n \le \lambda$. That is to say, the function is maximized when $\lambda = \max(x_1, x_2, \dots, x_n)$. Therefore, using MLE, we can obtain $\lambda = \max(x_1, x_2, \dots, x_n)$.

7.

(a)

$$EX = \int_{-\infty}^{\infty} xp(x)dx$$

$$= \int_0^{\infty} x \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} dx$$

$$= \frac{1}{\Gamma(\alpha)} \int_0^{\infty} (x\beta)^\alpha e^{-\beta x} dx$$

$$= \frac{1}{\beta\Gamma(\alpha)} \int_0^{\infty} (\beta x)^\alpha e^{-\beta x} d(\beta x)$$

$$= \frac{1}{\beta\Gamma(\alpha)} \underbrace{\int_0^{\infty} t^\alpha e^{-t} dt}_{t=\beta x}$$

$$=\frac{1}{\beta\Gamma(\alpha)}\Gamma(\alpha+1)$$

$$=\frac{1}{\beta\Gamma(\alpha)}\alpha\Gamma(\alpha)$$

$$=\frac{\alpha}{\beta}$$

(b)

Based on (a), we have $(EX)^2=\frac{\alpha^2}{\beta^2}$, next we compute $E(X^2)$

Similar to (a),

$$E(X^2)=\int_{-\infty}^{\infty}x^2p(x)dx$$

$$=\int_0^{\infty}x^2\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}dx$$

$$=\frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^{\infty}x^{\alpha+1}e^{-\beta x}dx$$

$$=\frac{\beta^\alpha}{\Gamma(\alpha)}\frac{1}{\beta^{\alpha+2}}\int_0^{\infty}(\beta x)^{\alpha+1}e^{-\beta x}d(\beta x)$$

$$=\frac{1}{\Gamma(\alpha)\beta^2}\underbrace{\int_0^{\infty}t^{\alpha+1}e^{-t}dt}_{t=\beta x}$$

$$=\frac{1}{\Gamma(\alpha)\beta^2}\Gamma(\alpha+2)$$

$$=\frac{1}{\Gamma(\alpha)\beta^2}(\alpha+1)\alpha\Gamma(\alpha)$$

$$=\frac{(\alpha+1)\alpha}{\beta^2}$$

Hence we can obtain variance

$$Var(X)=E(X^2)-(EX)^2$$

$$= \frac{(\alpha + 1)\alpha}{\beta^2} - \frac{\alpha^2}{\beta^2}$$

$$= \frac{\alpha}{\beta^2}$$

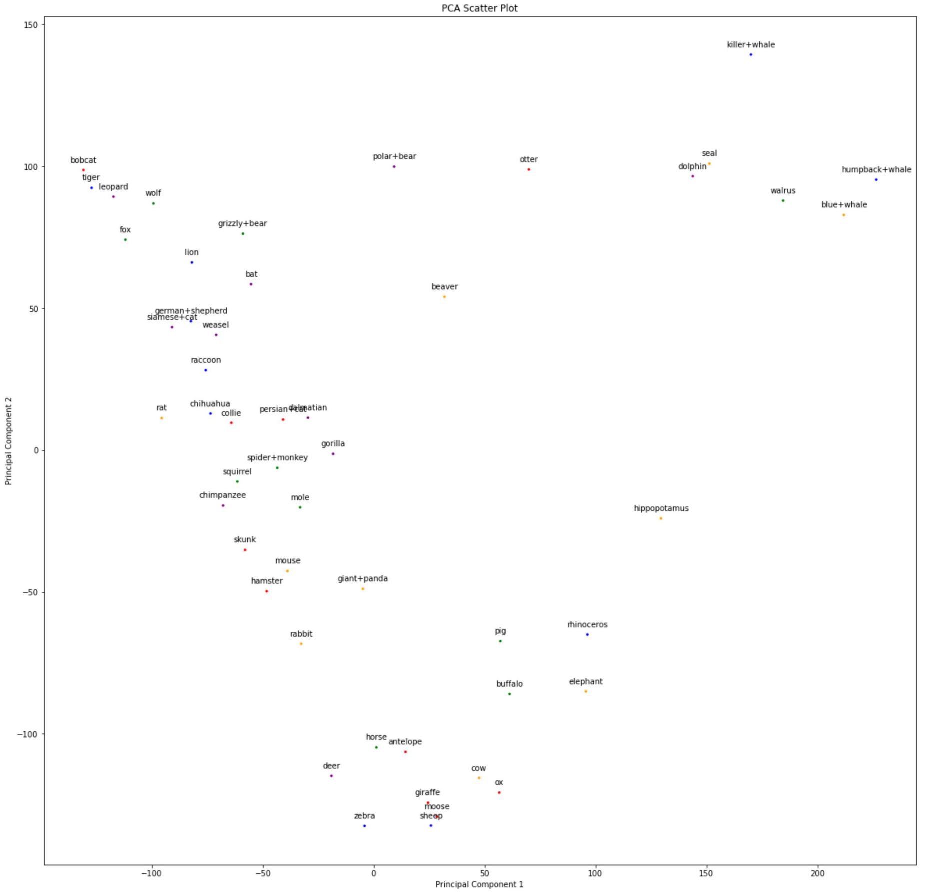
CSE291 HW4

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
import scipy
from pylab import rcParams
from scipy.cluster.hierarchy import linkage, dendrogram
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

In [2]:
```python
np.random.seed(1104)
predicates = np.loadtxt('predicate-matrix-continuous.txt')
animals = np.loadtxt('classes.txt', dtype = 'str')[:, 1]
predicates
```

Out[2]:
```
array([[-1.  , -1.  , -1.  , ...,  2.35,  9.7 ,  8.38],
       [39.25,  1.39,  0.  , ..., 58.64, 20.14, 11.39],
       [83.4 , 64.79,  0.  , ..., 15.77, 13.41, 15.42],
       ...,
       [63.57, 43.1 ,  0.  , ..., 35.95, 28.26,  5.  ],
       [55.31, 55.46,  0.  , ...,  5.04, 18.89, 72.99],
       [10.22, 21.53, 27.73, ...,  3.96, 14.05, 37.98]])
```

In [3]:
```python
pca = PCA(n_components=2)
pca.fit(predicates)
transformed = pca.transform(predicates)
rcParams['figure.figsize'] = 20,20
colors = ['red', 'green', 'blue', 'orange', 'purple']
colors = [colors[i%5] for i in range(len(transformed))]
plt.scatter(transformed[:, 0], transformed[:, 1], s=5,c=colors)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Scatter Plot')
for i, label in enumerate(animals):
    plt.annotate(label, (transformed[:, 0][i], transformed[:, 1][i]), textcoords="offset points", xytext=(0, 10), ha='center')

plt.show()
```

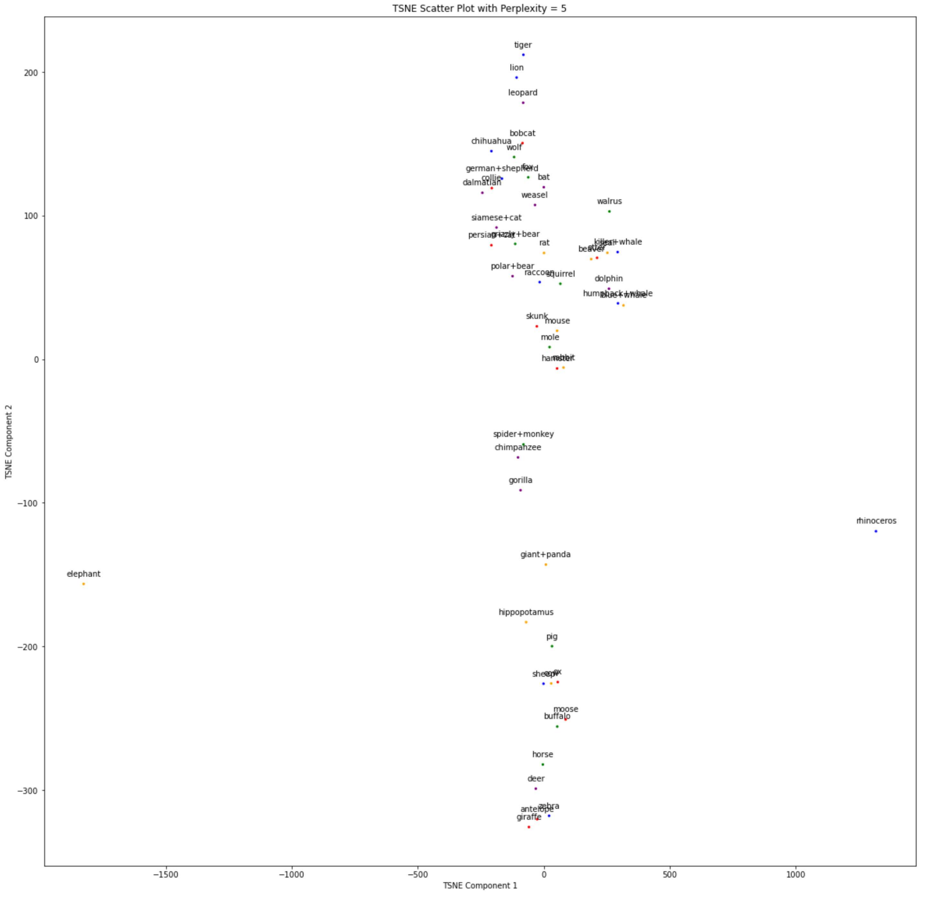


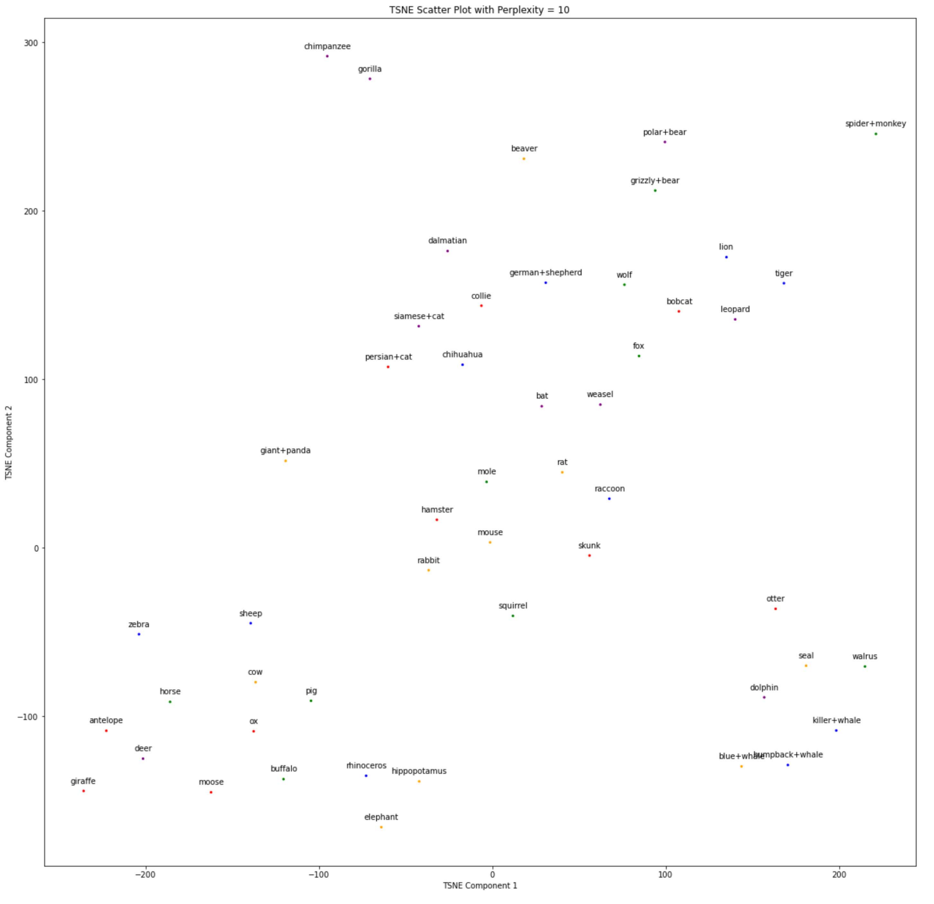
In [4]:
```python
def getFigures(theTransformedArray,perplexity):
    rcParams['figure.figsize'] = 20,20
    colors = ['red', 'green', 'blue', 'orange', 'purple']
    colors = [colors[i%5] for i in range(len(theTransformedArray))]
    plt.scatter(theTransformedArray[:, 0], theTransformedArray[:, 1], s=5,c=colors)
    plt.xlabel('TSNE Component 1')
    plt.ylabel('TSNE Component 2')
    plt.title('TSNE Scatter Plot with Perplexity = {}'.format(perplexity))
    for i, label in enumerate(animals):
        plt.annotate(label, (theTransformedArray[i][0], theTransformedArray[i][1]),
                     textcoords="offset points", xytext=(0, 10), ha='center')

    plt.show()
```

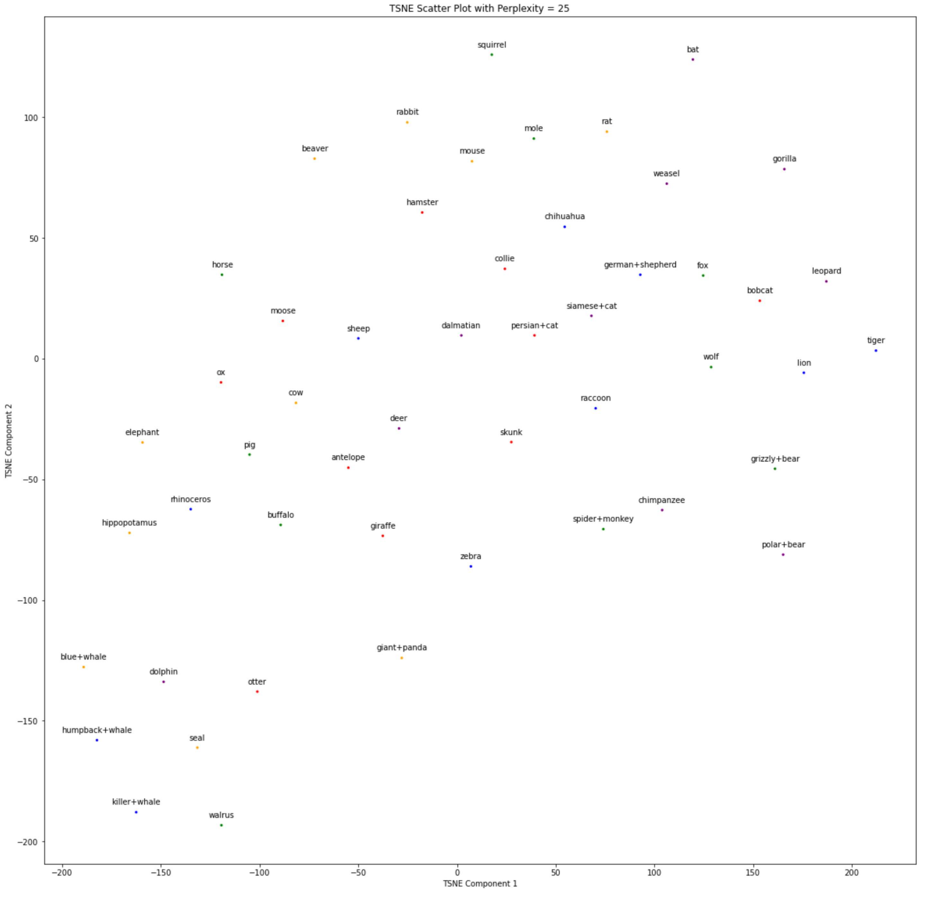
In [5]:
```
from sklearn.manifold import TSNE
Z = TSNE(n_components=2, perplexity=10.0).fit_transform(predicates)
Z
```

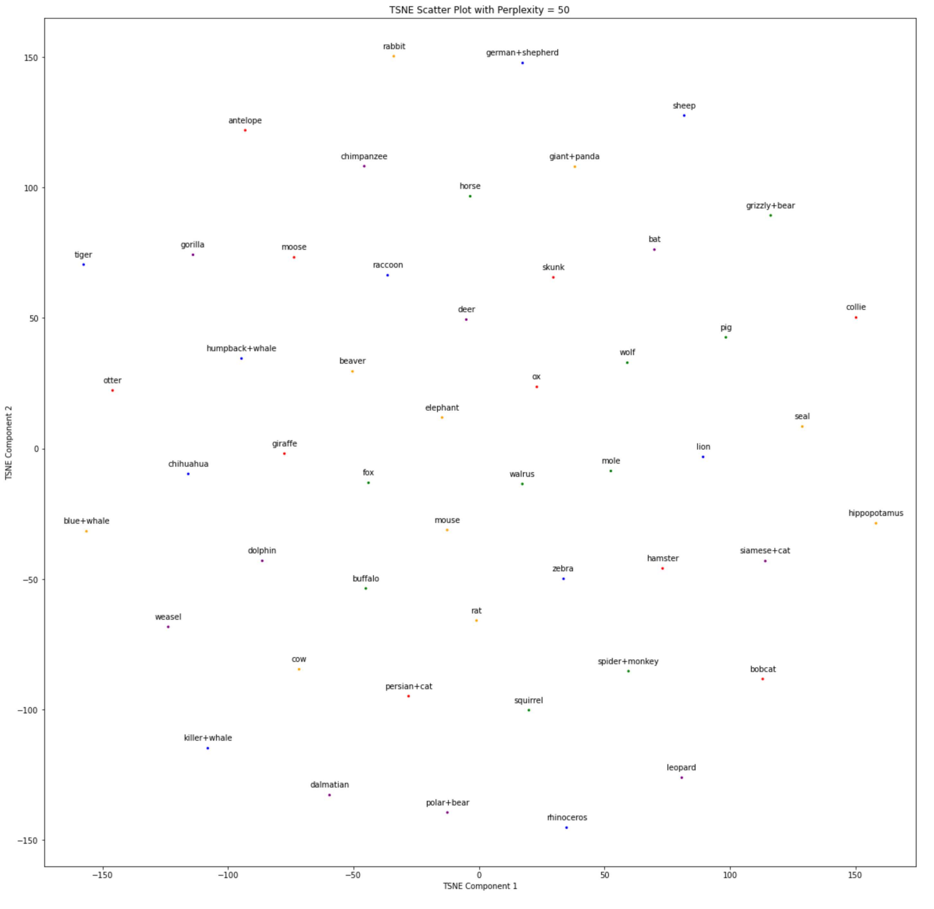
Out[5]:
```
array([[ 4.23630714e+01, -1.06522774e+02],
       [-1.24546585e+02,  9.12171783e+01],
       [ 1.12208488e+02,  7.60134201e+01],
       [ 5.04211845e+01,  7.13666611e+01],
       [-6.72778473e+01,  8.87574310e+01],
       [-5.27503471e+01,  4.09649658e+01],
       [ 6.71104355e+01, -9.92286377e+01],
       [-9.33370819e+01,  6.73102112e+01],
       [ 1.26584908e+02,  5.78591156e+01],
       [-6.89619751e+01,  4.79043045e+01],
       [-5.28704948e+01, -3.20122576e+00],
       [-9.88473511e+01, -2.26017323e+01],
       [-1.70222839e+02,  6.37803612e+01],
       [ 1.02078789e+02, -1.83318653e+01],
       [-1.52390213e+02,  5.51973000e+01],
       [ 1.17887505e+02, -8.19837952e+01],
       [-3.79082451e+01, -7.32223969e+01],
       [ 1.12000381e+02,  5.42657433e+01],
       [ 8.49601212e+01, -2.80779991e+01],
       [-1.68484764e+01, -6.78641357e+01],
       [ 9.68777542e+01, -7.19111023e+01],
       [-1.15239006e+02,  4.46380272e+01],
       [ 9.37096405e+01, -9.82254715e+01],
       [ 8.83803177e+01,  7.60931320e+01],
       [-2.53024502e+01, -8.14741974e+01],
       [-6.43882980e+01, -2.98645897e+01],
       [-9.19007111e+01, -5.29644241e+01],
       [ 1.01224388e+02, -3.54696960e+01],
       [-6.68618317e+01, -4.61969681e+01],
       [-8.61850357e+01,  2.24386215e+01],
       [ 4.52877731e+01, -8.64527817e+01],
       [-1.34811630e+02,  4.05489235e+01],
       [-5.34273872e+01,  6.19213638e+01],
       [-9.35970306e+01,  2.19650164e-01],
       [-1.05263313e+02,  2.46558666e+01],
       [ 6.57934570e+01,  7.30103607e+01],
       [ 1.10605911e+02, -5.96822128e+01],
       [ 8.22109680e+01, -1.22795380e+02],
       [ 1.87288227e+01, -3.01533012e+01],
       [ 5.44225121e+01, -1.14315697e+02],
       [-1.32283005e+02,  5.97370796e+01],
       [ 8.14108505e+01, -5.93077736e+01],
       [-1.52814041e+02,  7.47558670e+01],
       [-8.10489120e+01, -2.96686363e+01],
       [-1.21078209e+02,  1.07387787e+02],
       [-7.17005844e+01,  6.87122192e+01],
       [ 9.70652084e+01,  9.31534576e+01],
       [-7.48155899e+01,  2.43047029e-02],
       [ 8.24585037e+01, -7.88795319e+01],
       [ 9.20815430e+01,  5.92047958e+01]], dtype=float32)
```

In [6]:
```
Z_list=[]
for perplexity in [5,10,25,50]:
    Z = TSNE(n_components=2, perplexity=perplexity).fit_transform(predicates)
    Z_list.append(Z)
    getFigures(Z,perplexity)
```




TSNE Scatter Plot with Perplexity = 10
TSNE Component 1
TSNE Component 2
chimpanzee
gorilla
spider+monkey
polar+bear
beaver
grizzly+bear
dalmatian
lion
german+shepherd
wolf
tiger
collie
bobcat
siamese+cat
leopard
persian+cat
chihuahua
fox
bat
weasel
giant+panda
rat
mole
raccoon
hamster
mouse
skunk
rabbit
otter
squirrel
zebra
sheep
seal
walrus
cow
horse
pig
dolphin
antelope
ox
killer+whale
deer
blue+whale
humpback+whale
buffalo
rhinoceros
hippopotamus
giraffe
moose
elephant


TSNE Scatter Plot with Perplexity = 25
TSNE Component 1
TSNE Component 2
squirrel
bat
rabbit
rat
mole
beaver
mouse
gorilla
weasel
hamster
chihuahua
collie
horse
german+shepherd
fox
leopard
bobcat
siamese+cat
moose
sheep
dalmatian
persian+cat
tiger
wolf
lion
ox
cow
raccoon
deer
elephant
skunk
pig
antelope
grizzly+bear
rhinoceros
chimpanzee
buffalo
hippopotamus
giraffe
spider+monkey
polar+bear
zebra
giant+panda
blue+whale
dolphin
otter
humpback+whale
seal
killer+whale
walrus


TSNE Scatter Plot with Perplexity = 50
TSNE Component 1
TSNE Component 2
rabbit
german+shepherd
sheep
antelope
chimpanzee
giant+panda
horse
grizzly+bear
bat
tiger
gorilla
moose
raccoon
skunk
deer
collie
pig
humpback+whale
beaver
wolf
otter
ox
elephant
seal
giraffe
lion
chihuahua
fox
walrus
mole
blue+whale
mouse
hippopotamus
dolphin
hamster
siamese+cat
zebra
buffalo
weasel
rat
cow
spider+monkey
bobcat
persian+cat
squirrel
killer+whale
leopard
dalmatian
polar+bear
rhinoceros

In [7]:

```
def getDistanceList(aList):
    theDistanceList=[]
    for i in range(len(aList)-1):
        for j in range(i+1,len(aList)):
            theDistanceList.append(np.linalg.norm(aList[i] - aList[j]))

    return theDistanceList




def getDIstortionForTSNE(D,Z_list):
    for perplexity,Z in zip([5,10,25,50],Z_list):
        D_hat_TSNE = getDistanceList(Z)
        c2 = (sum(D)/len(D))/(sum(D_hat_TSNE)/len(D_hat_TSNE))
        delta_TSNE = [max(c2*j/i,i/(c1*j)) for i,j in zip(D,D_hat_TSNE)]
        print("distortion of TSNE with perplexity {} is {}".format(perplexity,sum(delta_TSNE)/len(delta_TSNE)))

D = getDistanceList(predicates)
D_hat_PCA = getDistanceList(transformed)

c1 = (sum(D)/len(D))/(sum(D_hat_PCA)/len(D_hat_PCA))


delta_PCA = [max(c1*j/i,i/(c1*j)) for i,j in zip(D,D_hat_PCA)]

print("distortion of PCA is {}".format(sum(delta_PCA)/len(delta_PCA)))
getDIstortionForTSNE(D,Z_list)
```

```
distortion of PCA is 1.8313495400958386
distortion of TSNE with perplexity 5 is 1.3349521969171132
distortion of TSNE with perplexity 10 is 1.2734444709416424
distortion of TSNE with perplexity 25 is 1.3964359251993639
distortion of TSNE with perplexity 50 is 1.6808162438567222
```

from what I've seen, the best is TSNE with perplexity 10 which acheives the lowest distortion

In [ ]: