A Tiny Taste of Machine Learning

All ML Methods Require

- Representation of the features
- Distance metric for feature vectors
- Objective function and constraints
- Optimization method for learning the model
- Evaluation method

Distance Between Vectors

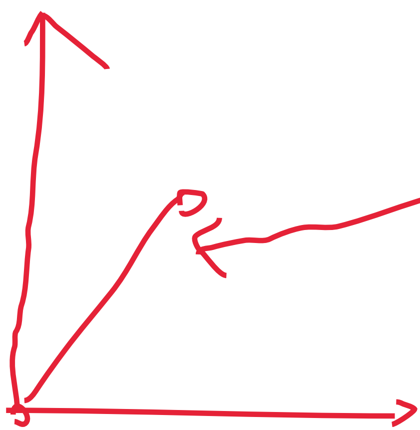
- Minkowski metric

p-norm

$$dist(X1,X2,p) = \boxed{\left(\sum_{k=1}^{len} abs(X1_k - X2_k)^p\right)^{1/p}}$$

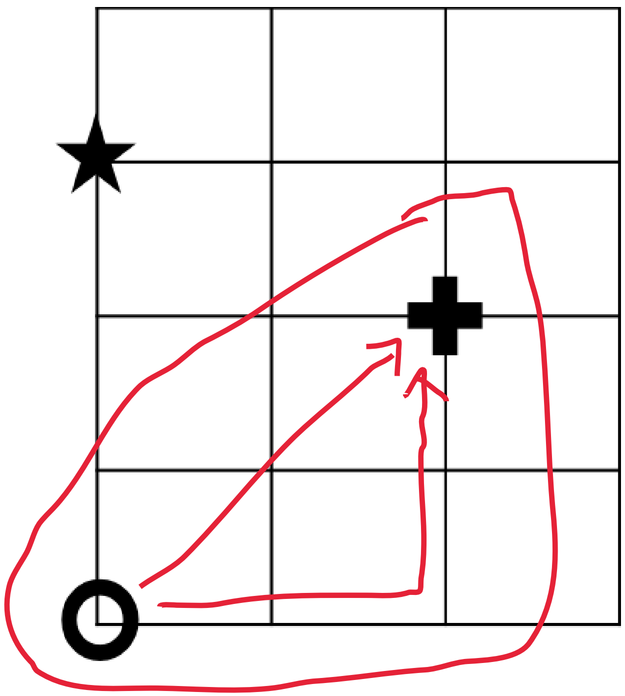
p = 1: Manhattan Distance 1-norm

P = 2: Euclidean Distance 2-norm



Manhattan (1902)

Which one is nearer?



Euclid-d: sqrt(8)

Manhattan-d: 3

An Example

Features **Label**

Name	Egg-laying	Scales	Poisonous	Cold-blooded	Number legs	Reptile
Cobra	1	1	1	1	0	1
Rattlesnake	1	1	1	1	0	1
Boa constrictor	0	1	0	1	0	1
Chicken	1	1	0	1	2	0
Guppy	0	1	0	0	0	0
Dart frog	1	0	1	0	4	0
Zebra	0	0	0	0	4	0
Python	1	1	0	1	0	1
Alligator	1	1	0	1	4	1

Distance Matrix

	cobra	rattlesnake	boa constrictor	chicken	guppy	dart frog	zebra	python	alligator
cobra	--	0.0	1.414	2.236	1.732	4.243	4.472	1.0	4.123
rattlesnake	0.0	--	1.414	2.236	1.732	4.243	4.472	1.0	4.123
boa constrictor	1.414	1.414	--	2.236	1.0	4.472	4.243	1.0	4.123
chicken	2.236	2.236	2.236	--	2.449	2.646	2.646	2.0	2.0
guppy	1.732	1.732	1.0	2.449	--	4.359	4.123	1.414	4.243
dart frog	4.243	4.243	4.472	2.646	4.359	--	1.414	4.359	1.732
zebra	4.472	4.472	4.243	2.646	4.123	1.414	--	4.359	1.732
python	1.0	1.0	1.0	2.0	1.414	4.359	4.359	--	4.0
alligator	4.123	4.123	4.123	2.0	4.243	1.732	1.732	4.0	--

Producing the Distance Matrix

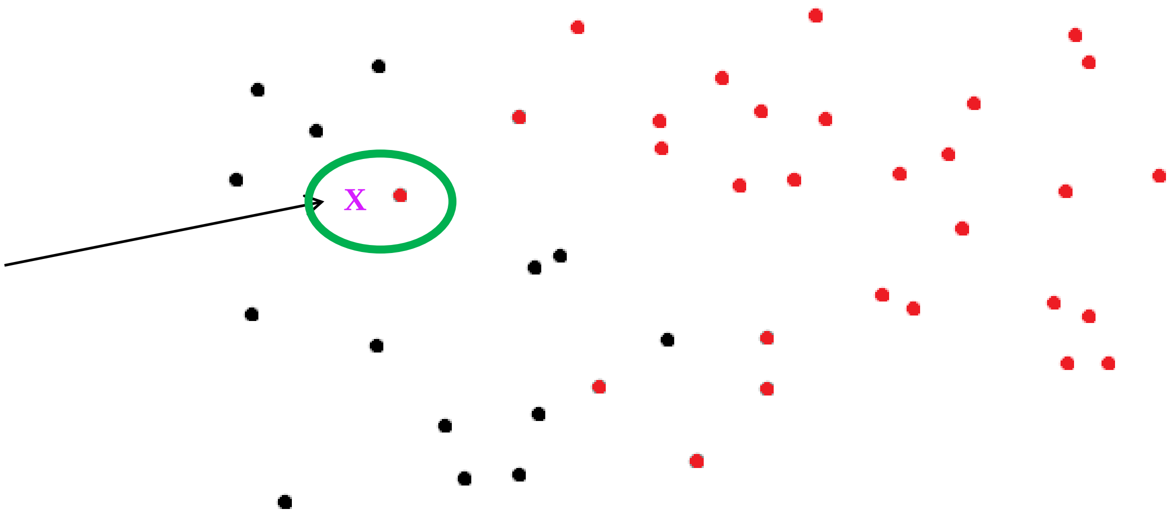
```
columnLabels = []
for a in animals:
    columnLabels.append(a.getName())
rowLabels = columnLabels[:]
tableVals = []
#Get distances between pairs of animals
#For each row
for a1 in animals:
    row = []
    #For each column
    for a2 in animals:
        if a1 == a2:
            row.append('--')
        else:
            distance = a1.distance(a2)
            row.append(str(round(distance, precision)))
    tableVals.append(row)
```

Producing the Table

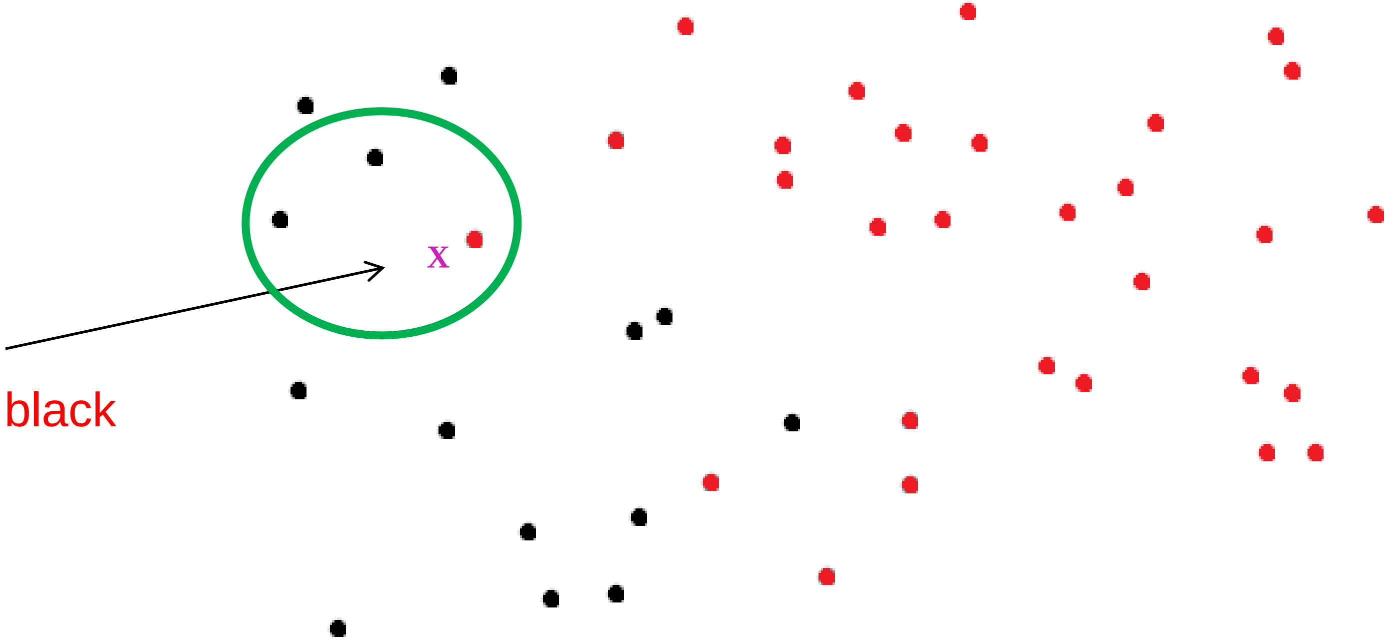
```
#Produce table
    table = pylab.table(rowLabels = rowLabels,
                        colLabels = columnLabels,
                        cellText = tableVals,
                        cellLoc = 'center',
                        loc = 'center',
                        colWidths = [0.138]*len(animals))
    table.auto_set_font_size(False)
    table.set_fontsize(10)
    table.scale(1, 2.5)
    pylab.axis('off')
    pylab.savefig('distances')
```

Using Distance Matrix for Classification

- Simplest approach is probably nearest neighbor
- Remember training data
- When predicting the label of a new example
  - Find the nearest example in the training data
  - Predict the label associated with that example



K-nearest Neighbors



Advantages and Disadvantages of KNN

- Advantages
  - Learning fast, no explicit training
  - No theory required
  - Easy to explain method and results
- Disadvantages
  - Memory intensive and predictions can take a long time
    - Are better algorithms than brute force
  - No model to shed light on process that generated data
- Another method, logistic regression, covered in textbook

An Example of KNN

Features **Label**

1—4 dis-proportionate places

Name	Egg-laying	Scales	Poisonous	Cold-blooded	Number legs	Reptile
Cobra	1	1	1	1	0	1
Rattlesnake	1	1	1	1	0	1
Boa constrictor	0	1	0	1	0	1
Chicken	1	1	0	1	2	0
Guppy	0	1	0	0	0	0
Dart frog	1	0	1	0	4	0
Zebra	0	0	0	0	4	0
Python	1	1	0	1	0	1
Alligator	1	1	0	1	4	1

1/0

KNN with k = 3

	cobra	rattlesnake	boa constrictor	chicken	guppy	dart frog	zebra	python	alligator
cobra	--	0.0	1.414	2.236	1.732	4.243	4.472	1.0	4.123
rattlesnake	0.0	--	1.414	2.236	1.732	4.243	4.472	1.0	4.123
boa constrictor	1.414	1.414	--	2.236	1.0	4.472	4.243	1.0	4.123
chicken	2.236	2.236	2.236	--	2.449	2.646	2.646	2.0	2.0
guppy	1.732	1.732	1.0	2.449	--	4.359	4.123	1.414	4.243
dart frog	4.243	4.243	4.472	2.646	4.359	--	1.414	4.359	1.732
zebra	4.472	4.472	4.243	2.646	4.123	1.414	→ non raptile		
python	1.0	1.0	1.0	2.0	1.414	4.359	raptile		
alligator	4.123	4.123	4.123	2.0	4.243	1.732			

not raptile

An Example of KNN

Features **Label**

Name	Egg-laying	Scales	Poisonous	Cold-blooded	Number legs	Reptile
Cobra	1	1	1	1	0	1
Rattlesnake	1	1	1	1	0	1
Boa constrictor	0	1	0	1	0	1
Chicken	1	1	0	1	2	0
Guppy	0	1	0	0	0	0
Dart frog	1	0	1	0	4	0
Zebra	0	0	0	0	4	0
Python	1	1	0	1	0	1
Alligator	1	1	0	1	4	1

KNN with k = 3

binary representation of leg

	cobra	rattlesnake	boa constrictor	chicken	guppy	dart frog	zebra	python	alligator
cobra	--	0.0	1.414	2.236	1.732	1.732	2.236	1.0	1.414
rattlesnake	0.0	--	1.414	2.236	1.732	1.732	2.236	1.0	1.414
boa constrictor	1.414	1.414	--	2.236	1.0	2.236	1.732	1.0	1.414
chicken	2.236	2.236	2.236	--	2.449	2.0	2.0	2.0	1.0
guppy	1.732	1.732	1.0	2.449	--	2.0	1.414	1.414	1.732
dart frog	1.732	1.732	2.236	2.0	2.0	--	1.414	2.0	1.732
zebra	2.236	2.236	1.732	2.0	1.414	1.414			
python	1.0	1.0	1.0	2.0	1.414	2.0			
alligator	1.414	1.414	1.414	1.0	1.732	1.732			

reptile √

A More General Approach: Scaling

- Z-scaling
  - Each feature has a mean of 0 & a standard deviation of 1
- Interpolation
  - Map minimum value to 0, maximum value to 1, and linearly interpolate

```
def zScaleFeatures(vals):
    """Assumes vals is a sequence of floats"""
    result = pylab.array(vals)
    mean = float(sum(result))/len(result)
    result = result - mean
    return result/stdDev(result)

def iScaleFeatures(vals):
    """Assumes vals is a sequence of floats"""
    minVal, maxVal = min(vals), max(vals)
    fit = pylab.polyfit([minVal, maxVal], [0, 1], 1)
    return pylab.polyval(fit, vals)
```