8. CLASSIFICATION, KNN, CURSE OF DIMENSIONALITY, SCALING, GRID SEARCH, PCA, IRIS, DIGITS

LEARNING OBJECTIVES

- Classification
- KNN
- Optimizing meta parameters with Grid search
- Visualization of higher dimensions
- Curse of dimensionality

REVIEW OF LESSON 7

LAST LESSON REVIEW

- Bias - Variance decomposition
- Cross validation
- Sampling, Boostrapping
- Strategies to deal with Overfitting and Underfitting

LAST SESSION

ANY QUESTIONS?

- How to reduce Bias / Underfitting?
- How to reduce Variance / Overfitting?

TODAY

CLASSIFICATION

CLASSIFICATION

REGRESSION

- Predict Continuous values

CLASSIFICATION

Predict a finite set of values aka labels

- Binary classification (sex of babies, survival on Titanic)
- Multiclass classification (user segmentation, flower type Iris dataset)
- Ordinal classification: classes are ordered (Grades of students, stars in user reviews, ...)

REGRESSION OR CLASSIFICATION?

Review the following situations and decide if each one is a regression problem, classification problem, or neither:

- Using the total number of explosions in a movie, predict if the movie is by JJ Abrams or Michael Bay.
- Determine how many tickets will be sold to a concert given who is performing, where, and the date and time.
- Given the temperature over the last year by day, predict tomorrow's temperature outside.
- Using data from four cell phone microphones, reduce the noisy sounds so the voice is crystal clear to the receiving phone.
- With customer data, determine if a user will return or not in the next 7 days to an e-commerce website.

REGRESSION TO CLASSIFICATION

How can regression models be used for classification?

REGRESSION TO CLASSIFICATION

How can regression models be used for classification?

=> Using Threshold(s)

```
if probability < 0.5:
    return A
else:
    return B
```

LET'S BUILD A CLASSIFIER!

Binary classifier to differentiate between class 0 and other classes

- Load the iris dataset and Boxplot the petal length and the petal width
- Write a simple function that returns 0 or 2 depending on the Petal length and a threshold
- Write a function that returns the ratio of properly predicted samples vs the total number of samples
- For different thresholds calculate the number of estimated positives (1s) over the true number of samples in that class

What happens to the accuracy if you were to look at sepal length for your simple classifier? Try it.

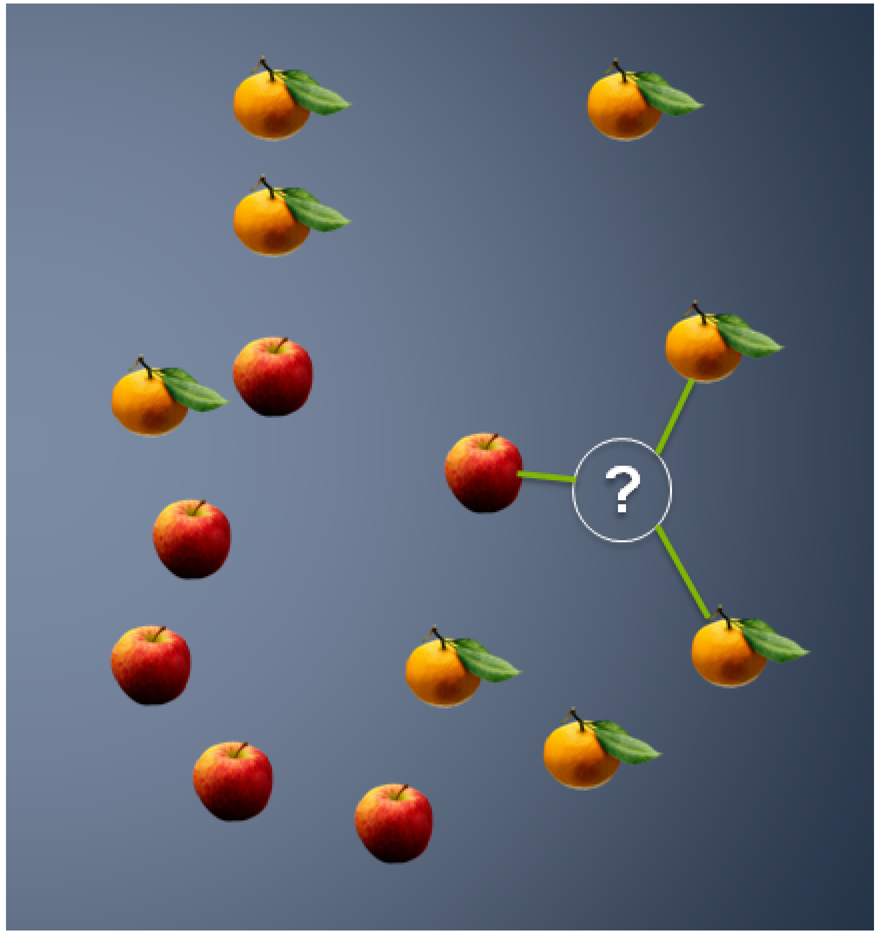
How would you use this simple classifier to expand to 3 classes

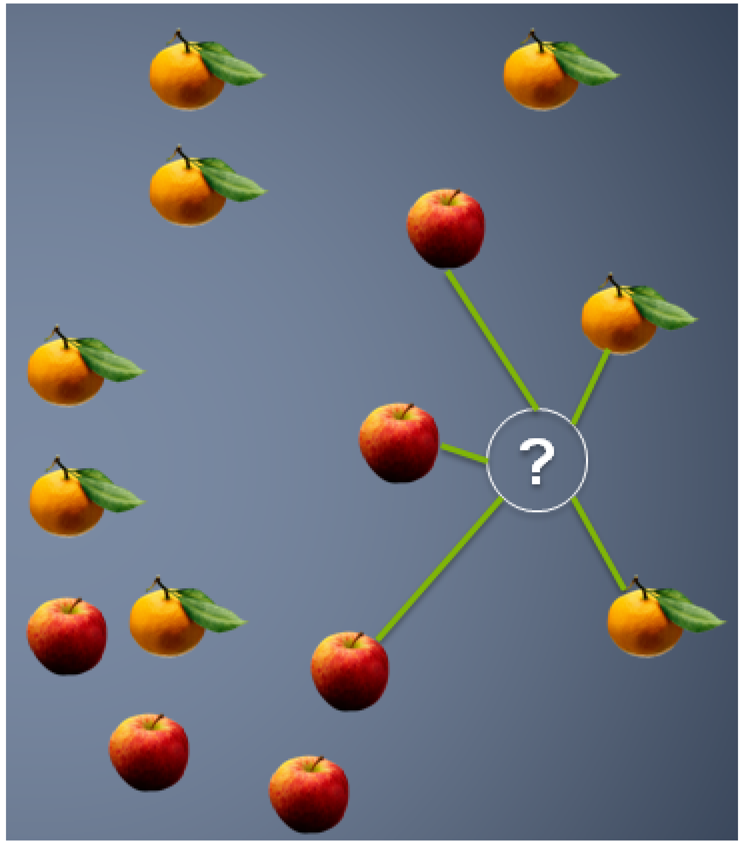
K NEAREST NEIGHBORS

You have a set of samples already labeled

1. For each new sample, calculate the *distance* to **all** other points.
2. Given those distances, pick the **K** closest samples.
3. Count the number of samples in each class
4. The new sample is classified as the class label with the largest number ("votes").

K NEAREST NEIGHBORS





DEMO: K NEAREST NEIGHBORS

- Load the iris dataset
- Shuffle it
- Split 100 samples for training and 50 for testing
- See how the accuracy on the test set evolves with K ranging from 2 to 20
- Plot the training error and the test error vs K

```
from sklearn import datasets, neighbors, metrics
import pandas as pd

iris = datasets.load_iris()
knn = neighbors.KNeighborsClassifier(n_neighbors=K)
knn.fit(iris.data, iris.target)
...
metrics.accuracy_score(y_test, y_hat )
```

YOUR TURN

Plot Accuracy for training and test error as a function of K

- set the seed
- Load iris dataset and shuffle it
- split X and y in train / test dataset (100, 50 samples)
- (loop over K) for each K
  - Initialize the classifier with K
  - Fit
  - Predict
  - print K and the accuracy score

Notebook 1 - solution

GRID SEARCH: SEARCHING FOR ESTIMATOR PARAMETERS

How do you find the best hyperparameter for your model?

- α for Ridge or Lasso
- K for K-NN
- ...

Grid search and Randomized Search allow you to specify a parameter space and find which value(s) of the hyperparameter(s) results in the best performance of your model.

GRID SEARCH: SEARCHING FOR ESTIMATOR PARAMETERS

METRIC

- Accuracy for classification
- $R^2|$ for regression
- Specify another metric via the *scoring* function

K-NN + GRID SEARCH

Now Find K with Grid Search on Fraudulent dataset

- fraud: cases
- student: is the person a student?
- balance and income

WALKTHROUGH

- knn = neighbors.KNeighborsClassifier()
- parameters = {'n_neighbors' : np.arange(2,20) }
- clf = GridSearchCV(knn, parameters)
- clf.best*estimator*, clf.best*params*, clf.best*score*

K-NN + GRID SEARCH

YOUR TURN

- Fraud as function of balance
- Looking at the accuracy, what's the best K?
- What's happening here?

OTHER KNN HYPER PARAMETERS

by default

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=5, p=2, weights='uniform')

MINKOWSKI DISTANCE

for 2 points $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$

The Minkowski distance is defined by

$$d(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \quad \text{with } p \geq 1$$

- For p = 1:Manhattan distance

$$d(X, Y) = \sum_{i=1}^{n} |x_i - y_i|$$

- For p = 2: Euclidian distance

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

GRID SEARCH WITH K AND DISTANCE

- parameters = {'n_neighbors' : np.arange(2,20), 'p': np.linspace(1,2,5)
  }
- What's the best distance parameter p, what's the best K?
- What's the next best score parameters?

PRE PROCESSING THE DATA WITH SCIKIT

preprocessing

- scaling
- normalizing
- remove linear correlation with PCA
- binarization
- encoding categorical features
- missing values

with

- fit
- transform

SCALING AND CENTERING

Let's scale and normalize the income and balance and see if that helps

MAKE CLASSIFICATION

make_classification

n_features = n_informative + n_redundant + n_repeated + noisy feature

- what's the influence of n_redundant
- what's the influence of noisy features,
- does repeating features help? why?
- what's the catch?

DIMENSIONALITY REDUCTION WITH PCA

PRINCIPAL COMPONENT ANALYSIS

Problem: Can't visualize n Dimensions

Goal: Project the data space to a 2 dimension space to visualize it

PCA: orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components

- by Sebastian Raschka
- In scikit

VISUALIZE HIGHER DIMENSIONS DATASETS

Let's use the digits dataset in scikit

```
digits = datasets.load_digits()
X = digits.data
y = digits.target
```

- Dimension of the feature space?
- use PCA to project on 2 dimensions

```
XX = pca.fit(X).transform(X)
```

- Plot with color as a target

```
plt.scatter(XX[:,0], XX[:,1], c = y, cmap = plt.cm.coo
```

L8 N4 PCA

The Curse Of Dimensionality

CURSE OF DIMENSIONALITY

- Create a set a classifications set with 100 samples and 5 classes
- Use only significant features (n_features = n_significant)
- Increase the number of features from 4 to 25

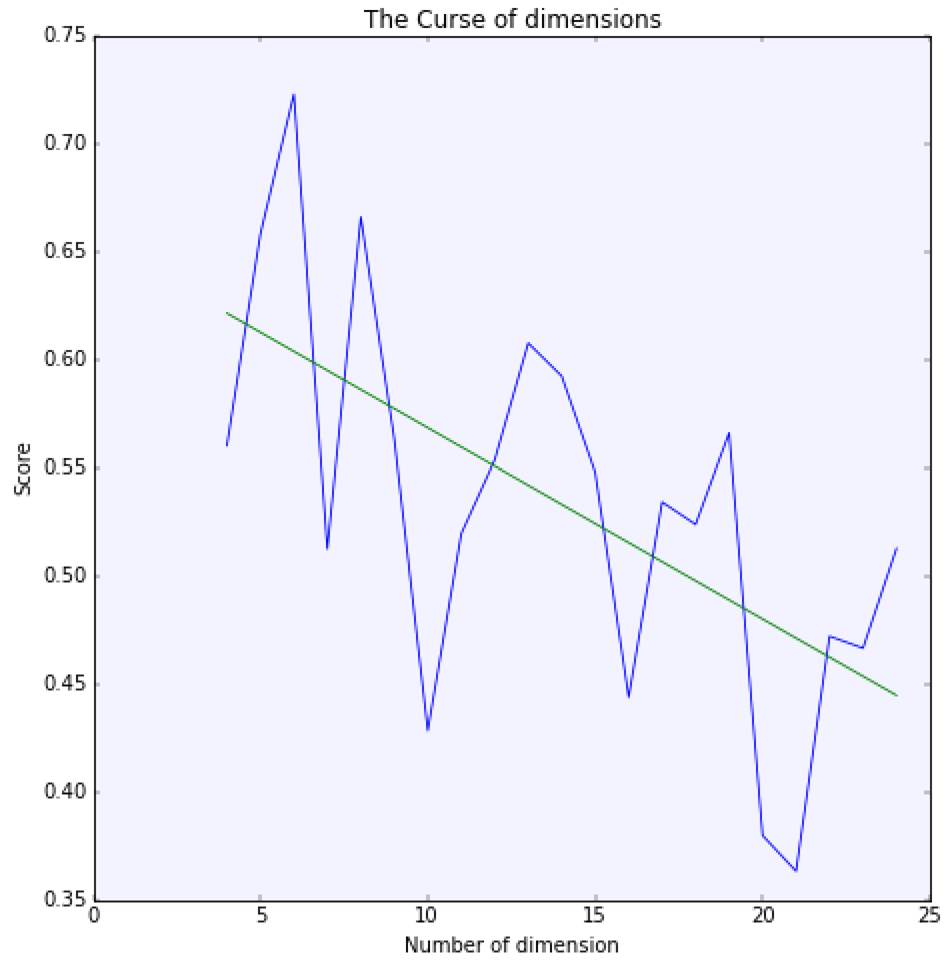
Plot the error vs the number of dimension.

What do you get?

Why?

L8 N5 - Curse of Dimensionality

CURSE OF DIMENSIONALITY



The more dimensions the more it is difficult for the classifier to measure the distance significantly

CURSE OF DIMENSIONALITY

You want your training dataset to cover 20% of the range of the feature space

- in 1D you need 20% of the population
- in 2D you need 45% of the population ($0.45^2 = 0.2$)
- in 3D you need 58% of the population ($0.58^3 = 0.2$) => overfitting

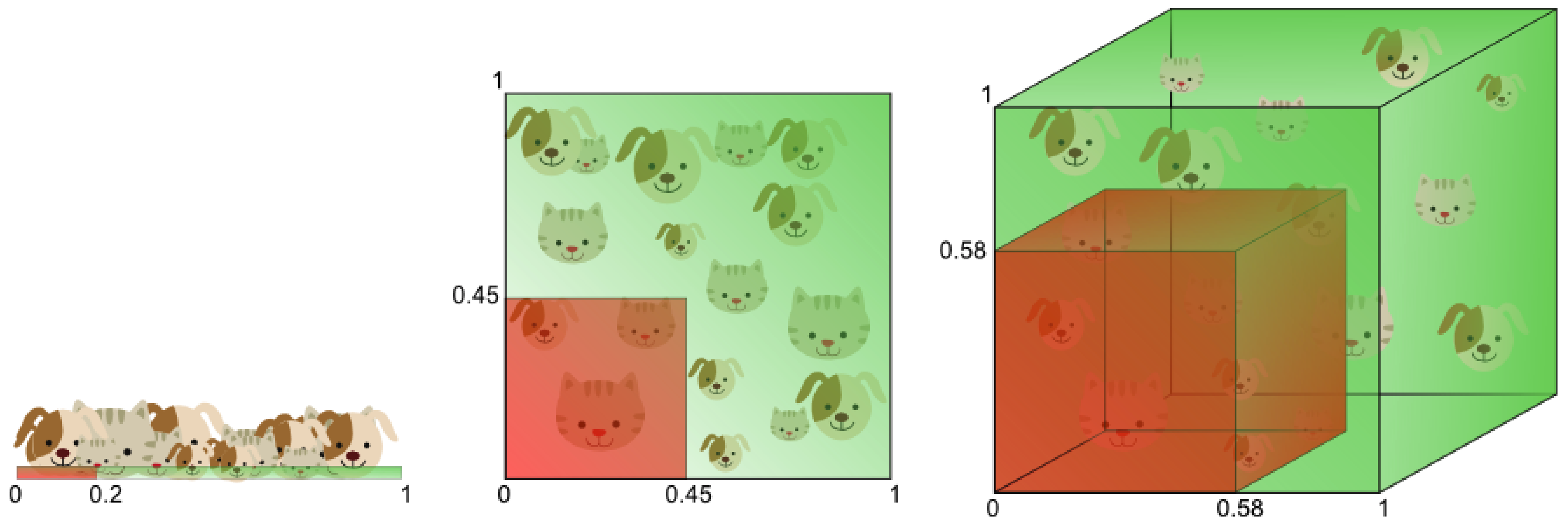


Figure 8. The amount of training data needed to cover 20% of the feature range grows exponentially with the number of dimensions.

CURSE OF DIMENSIONALITY

Sample density drops, data gets scarce, distance between points is less discriminative

Let's say your features all range from 0 to 5 (5 units). and you have 10 samples

- in 1D, density = 10 / 5 = 2 samples per unit
- in 2D, density = 10 / 25 = 0.5 samples per unit
- in 3D, density = 10 / 125 = 0.08 samples per unit

KNN

- Predict probability
- KNN for regression

http://scikit-learn.org/stable/auto_examples/neighbors/plot_regression.html

LINKS

- K-NN mathematicalmonk's channel
- Implementing your own k-nearest neighbour algorithm using Python
- Centering, scaling and k-Nearest Neighbours
- Scikit Classification Example
- Scikit Nearest Neighbors
- Classification in Python with scikit-learn
- The Curse of Dimensionality in classification