PHYS 498

Aiden Sirotkine

Spring 2025

Contents

Chapter 1

PHYS498

a whoooole lotta data analysis and fun stuff like that.

1.1 Clustering

group together sample data points that are a certain distance from each other

$$d(i,k) = \sum_{\text{features } i} (x_{ji} - x_{ki})^2$$

However, what if data points have different units?

ML algorithms are "unit-agnostic", which means they don't really care about units.

1.2 Whitening Transformation

$$x \to \frac{x-\mu}{\sigma}$$

it makes the mean 0 and the standard deviation 1

It just removes the white noise from a dataset.

a whitening transformation is a linear transformation that transforms a vector of random variables with a known covariance matrix into a set of new vararIables whose covariance is the identity matrix, meaning that they are uncorrelated and each have unit variance.

whitening the inputs is useful because machine learning likes standardized data.

1.3 Examples

you slam bags of apples together and columnated jets of walnuts come out

1.3.1 Atlas

We look at the random jets of energy and we cluster the data to figure out what particles are where.

1.3.2 Gamma Ray Bursts

We can look into the universe and see random bursts of gamma rays

We find clusters of bursts at the galactic plane and around the center of the galaxy.

We cluster the data and discover the Fermi Bubble.

They're produced by colliding neutron stars and collapsing normal stars into black holes.

1.4 K-means Clustering

fast and robust decent algorithm. Assume you data consists of roughly round clusters of roughly the same size.

a_fit = cluster.KMeans(n_clusters=2).fit(a_data)

1.5 Hyperparameters

parameters that have to be pre-set that determine how the data is going to be analyzed.

For example, the number of clusters that we want as a result from K-means

1.6 ML Algorithms

Maximize the goal functions given the data.

The goal function $\mathcal{L}$ of the KMeans algorithm is

$$\mathcal{L}(c_j) = \sum_{i=1}^{n} \sum_{c_j=i} |x_j = \mu_i|^2$$

where $c_j = 1$ if the sample j is assigned to cluster I or otherwise

$c_j = 0$ and

$$\mu_i = \sum_{c_j = i} x_j$$

1.6.1 Expectation-Maximization

real important for alot of algorithms but I don't fully understand it.

1.7 Curse of Dimensionality

r^D is the volume of a D-dimensional hypercube with each dimension subdivided by r partitions.

a 3×3 rubix cube has 27 mini cubes in it because 3^3

It basically means that the more dimension you have, the more cooked you are to process all of it.

The curse of dimensionality.

If we have 30 dimensional data, we need over 1 billion data points in order to get a sample in each partition.

If there's a functional dependence between demensions, you can use a transformation to get rid of it and then you're working with less dimensions which is a win.

In the slides, 500 dimensional data can get transformed into 2 dimensional data.

1.8 Linear Decompositions

solve the eigenvector eigenvalue problem and delete the unimportant vectors and boom removed the most useless dimensions.

Principle Component Analysis (PCA) was developed in 1901 and is basically just removing the smallest/least impactful eigvenvectors

1.8.1 Covariance Matrix

$$C = \frac{1}{N-1} X^T X$$

is an estimate of the true covariance matrix using the data X comprised of N samples that are D-dimensional.

M is matrix where each row is an eigenvector

Y is a matrix such that $X = YM$

Remove dimensions from D to d with the smallest eigenvalues.

Now you have a much smaller matrix with most of the same data.

1.8.2 Eigenmatrix of $X^T X$

$$M^T = M^{-1} \qquad MM^T = I$$

$$X^T X = M^T \Lambda M$$

Where Λ is the diagonal matrix of decreasing eigenvalues.

The resulting latent variables are not correlated to each other, which means

$$\rho(j,k) = \frac{Y_J \cdot Y_k}{|Y_j||Y_k|} \simeq 0$$

1.9 Factor Analysis

another good way to reduce data

1.9.1 Non-negative Matrix Factorization

another thing

1.10 Independent Component Analysis

another thing

These aren't really important but like depending on what data you're doing you might need more than PCA.

1.11 Kernel Functions

If you add a bunch of random ass dimensions to your data, you can observe a number of correlations that you would not have seen otherwise

$$\phi(x_0, x_1) = \begin{bmatrix} x_0^2 \\ x_0 x_1 \\ x_1 x_0 \\ x_1^2 \\ \sqrt{2c} x_0 \\ \sqrt{2c} x_1 \\ c \end{bmatrix}$$

This kernel function yields shenanigans

$$\phi(X_i) \cdot \phi(X_j) = (X_i + X_j + c)^2$$

This allows you to embed your data into higher dimensional space without actually using a bunch of math.

Our kernel function will be written as

$$K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$$

A kernel function is a similarity measure, since it measures the similarity between samples i and j, with identical values being maximal and orthogonal values being minimal.

This is known as the kernel trick.