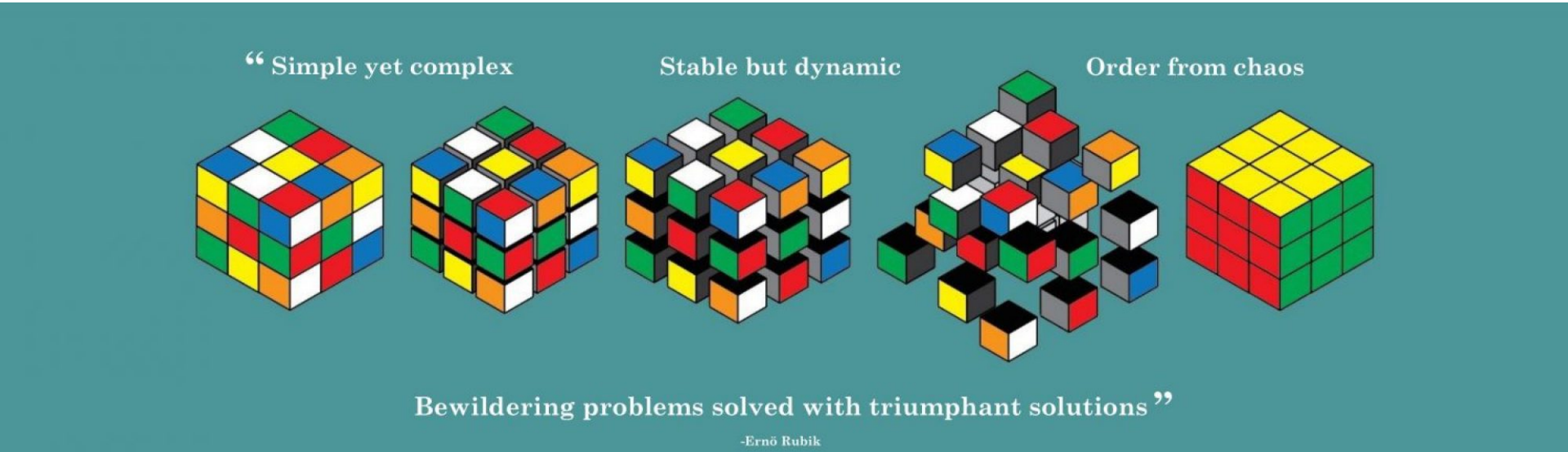
Applied Neuroscience

Columbia Science Honors Program

Fall 2016

Machine Learning and Neural Networks



Machine Learning and Neural Networks

Objective: Introduction to Machine Learning

Agenda:

1. JavaScript Tutorial for Handwritten Digit Recognition
2. Fundamentals of Machine Learning
   Principal Component Analysis
3. Tour of the Laboratory of Professor Paul Sajda

Three Layers of Web Design

1. **HTML for Content**
2. **CSS for Presentation**
3. **JavaScript for Behavior**

Advantages	Disadvantages
Runs Fast: JS does not need to contact server	Runs on Single Thread: JS has no multi-processing capabilities
Platform Independent	Read/Write Files: JS disables this for security reasons
Easy to Learn	Networking Applications
Increased Interactivity: interfaces react on user interactions	Code is not hidden: JS code always visible to client
Richer Interfaces: JS can be used for sliders	

JavaScript for Handwritten Digit Recognition

Euclidean Distance: distance between a pair of points in Euclidean space

The Euclidean distance provides a value that determines the similarity between two arrays of the same length.

Consider the following three arrays:

```
Array 1: [1, 2, 3]
Array 2: [1, 3, 2]
Array 3: [3, 2, 1]
```

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

How can we determine the arrays most similar to each other?

1. Compute Euclidean distance between Array 1 and Array 2
2. Compute Euclidean distance between Array 1 and Array 3
3. Compare: shorter distance represents greater similarity

Implement Euclidean Distance in JavaScript

```
ENCOG.MathUtil.euclideandistance=function(a1,a2,startIndex,len)
{ 'use strict';

    var result = 0, i, diff;
    for (i = startIndex; i < (startIndex + len); i += 1) {
        diff = a1[i] - a2[i];
        result += diff * diff;
    }
    return Math.sqrt(result);
};
```

Euclidean Distance can be used to create a simple optical character recognition.

JavaScript Handwritten Recognition Demo

JavaScript for Handwritten Digit Recognition

1. **Train program to recognize digits.**
   What do we use to train algorithms?
2. **Draw a digit.**
3. **Down-sample high resolution characters onto a 5x8 grid**
   What is down-sampling?
   At this step, the character drawn becomes a matrix array.
4. **Find the array *most similar* to the digit drawn in Step 2.**
   What criterion is used to measure similarity between arrays?

Implement Euclidean Distance in JavaScript

```
var c, data, sum, i, delta

for(c in charData)
{
    data = charData[c];
    //Now we will recognize the letter drawn using
    //Euclidean Distance

sum = 0;
for(var i=0; i<data.length; i++)
{
    delta = data[i] - downSampleData[i];
    sum = sum + (delta*delta);
}
    sum = Math.sqrt(sum);

    //Smallest Euclidean Distance is the char

if( sum<bestScore || bestChar==‘??’)
{
    bestScore = sum;
    bestChar = c;
}
}
```

Principal Component Analysis

Objective: Identify patterns in data

By reducing a ***d*-**dimensional dataset by projecting it onto a ***(k)*-**dimensional subspace (where $k < d$), we increase computational efficiency while retaining most of the information.

What is the size of k that represents the data well?

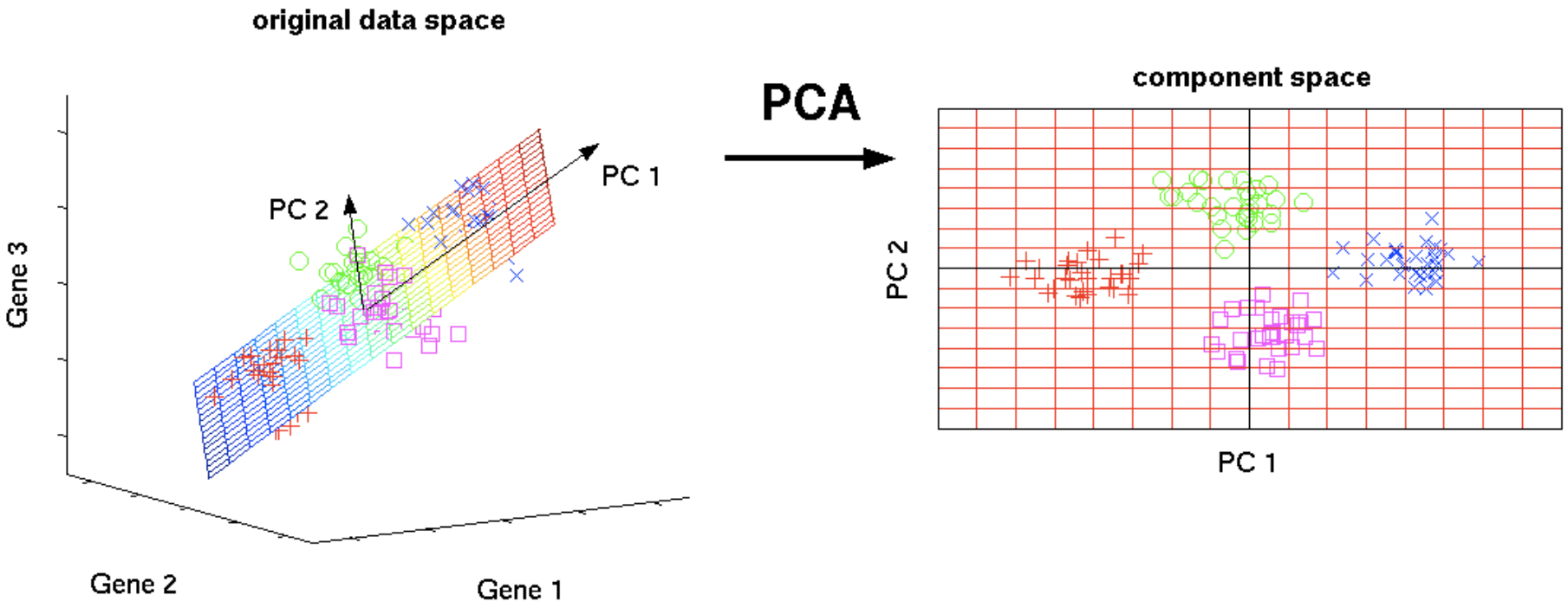
Analysis:

Find the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace while retaining most of the information.

Principal Component Analysis

1. Standardize the Data
2. Obtain the Eigenvectors and Eigenvalues:
    1. *Covariance matrix: How do we determine the similarity of arrays?*
    2. *Singular Vector Decomposition*
3. Sort eigenvalues in descending order and choose the ***k*** eigenvectors that correspond to the ***k*** largest eigenvalues where ***k*** is the number of dimensions of the new feature subspace ***(k ≤ d)***
4. Construct the projection matrix **W** from the selected ***k*** eigenvectors
5. Transform the original dataset **X** via **W** to obtain the *k*-dimensional feature subspace **Y.**

Principal Component Analysis



Principal Component Analysis uses an orthogonal transformation to convert a set of observations of *possibly correlated* variables into a set of values of *linearly uncorrelated* variables called **principal components.**