10th Inverted CERN School of Computing

6 – 8 March 2017
CERN, IT Amphitheatre (31/3-004)

Live webcast, slides & recording at
https://indico.cern.ch/e/iCSC-2017

Lecturers

Georgios BITZES | CERN
Daniel CAMPORA PEREZ | CERN & Universidad de Sevilla
Michael DAVIS | CERN
Daniel LANZA | CERN
Lorena LOBATO PARDAVILA | CERN & Universidad de Oviedo
Eamonn MAGUIRE | Pictet Asset Management
Some might say this rather lovely picture has “nothing to do with computing”. But if a certain publisher of programming books can put pictures of random animals on their stuff, then we are going to put cool pictures of stuff at CERN on our stuff 😊
I warmly welcome you all, lecturers and attendees alike, to this Inverted CERN School of Computing (iCSC) 2017. There are three reasons why I feel particularly happy and privileged to be writing these words.

First, this year we have a very rich program, based on proposals received from students of the main CSC 2016 (Mol, Belgium) and the Thematic CSC 2016 (Split, Croatia). It consists of 11 hours of lectures and one hour of hands-on workshop (a novelty!). Topics include Artificial Intelligence; Distributed Consensus & Fault Tolerance; Anomaly Detection; Evolutionary Computation; VM Image Management; and Data Visualizations. I’m sure you will find them both relevant and interesting!

Secondly, this is already the 10th edition of the Inverted CSC. It was founded by one of my predecessors, Francois Fluckiger, in 2005 (initially, the Inverted School did not take place every year). His idea was give the floor to former students of the main CERN School of Computing, so that they could share their knowledge and expertise with their colleagues. The fact that we've reached the 10th edition this year, and that we actually received more proposals for lectures than we were able to accommodate, proves that this idea is still valid.

And finally, on a more personal note, the Inverted CSC remains close to my heart. I was one of the lecturers of its first edition back in 2005, giving a lecture about software security - even though I worked on something else at that time. I still remember how this experience had pushed me to develop further my knowledge and passion in security, and - consequently – heavily influenced my professional life. I hope this year’s iCSC will have a similar positive impact on the careers of the lecturers!

I wish to thank the lecturers for their significant work put in preparing the lectures; the mentors for their input and feedback; Cath Noble (the School’s Administrative Manager) for the support behind the scenes; and finally, you - the attendees - for your interest.

Please join me in enjoying this great learning and knowledge-sharing experience!

Sebastian Łopieński
Director
CERN School of Computing
Welcome

Contents

Core team

Biographies & lecture abstracts

Timetable

Appendix : lecture slides

Daniel Campora Perez
Let your machine do the learning - Lecture 1 .............................. A.1
Let your machine do the learning - Lecture 2 .............................. A.14

Michael Davis
Algorithms for Anomaly Detection - Lecture 1 .............................. A.26
Algorithms for Anomaly Detection - Lecture 2 .............................. A.44

Daniel Lanza
Applying natural evolution for solving computational problems - Lecture 1......... A.62
Applying natural evolution for solving computational problems - Lecture 2......... A.70

Georgios Bitzes
Distributed consensus and fault tolerance - Lecture 1........................ A.77
Distributed consensus and fault tolerance - Lecture 2........................ A.88

Lorena Lobato Pardavila
Virtual Machine Images Management in Cloud Environments.................. A.100

Eamonn Maguire
Creating Effective Data Visualizations - Lecture 1........................ A.110
Creating Effective Data Visualizations - Lecture 2........................ A.139
CORE TEAM

Director
Sebastian ŁOPIENSKI

Administration
Cath NOBLE

Mentors
Olivier COUET

Nikos KASIOUMIS

Sebastian ŁOPIENSKI

Alberto PACE

Danilo PIPARO

Sebastien PONCE

Ivica PULJAK

Enric TEJEDOR

The inverted School of Computing (iCSC) is part of the annual series of schools organized by the CERN School of Computing http://cern.ch/csc
Every lecturer presenting here at the iCSC 2017 has been specially selected to advance their ideas, to further develop CSC-related themes, and to share and promote their knowledge.

All lecturers have been students at either CSC 2015 (Kavala, Greece), tCSC 2016 (Split, Croatia) or CSC 2016 (Mol, Belgium).

The lectures are broadcast via webcast and will be recorded. All slides, recordings etc can be found at https://indico.cern.ch/e/iCSC-2017

Georgios BITZES

CERN, Switzerland

I am currently a Fellow working on data storage at CERN. My main project right now is building a highly-available distributed database for the EOS namespace, built upon rocksdb and the raft consensus algorithm.

I received my BSc in Computer Science from the Department of Informatics and Telecommunications, University of Athens. My main interests include performance optimization, distributed computing, and algorithmic problems.

Mentors
Sebastian Łopiński, Enric Tejedor

Lecture
Distributed consensus and fault tolerance

In a world where clusters with thousands of nodes are becoming commonplace, we are often faced with the task of having them coordinate and share state. As the number of machines goes up, so does the probability that something goes wrong: a node could temporarily lose connectivity, crash because of some race condition, or have its hard drive fail.

What are the challenges when designing fault-tolerant distributed systems, where a cluster is able to survive the loss of individual nodes? In this lecture, we will discuss some basics on this topic (consistency models, CAP theorem, failure modes, byzantine faults), detail the raft consensus algorithm, and showcase an interesting example of a highly resilient distributed system, bitcoin.
Coming from Sevilla with a Computer Engineering degree and having spent some five years at CERN with various contracts, I'm currently a Doctoral Student, optimizing reconstruction algorithms in LHCb with an Artificial Intelligence twist.

I have many good questions, but I'm still searching for good answers!

Mentors
Sebastian Łopiński, Danilo Piparo

Lectures
Let your machine do the learning

The field of Artificial Intelligence, whose formal definitions go as back as the 40's, have recently gained a renowned interest in the community as more and more problems become amenable to be tackled by it. Even better, we are now available to try complex techniques in a very accessible manner, lowering the entrance admission to this cool club to just a couple of hours.

In this series of lectures, we are going to explore non-conventional techniques to solve long standing problems, coming from AI, from a pragmatic and up to date perspective. By the end of these lectures you should be able to get your hands dirty with exciting real examples. You should be able to identify what kind of problems are you dealing with, what tools does AI have in store for you and how to apply them in a straightforward way, with room for depth. Just enjoy longer coffee breaks as the machine works it out for you.
Michael DAVIS

CERN, Switzerland

Michael is a software engineer at CERN. He has spent the last three years developing part of the power control system for the LHC.

Michael holds a BSc. in Computer Science from Brunel University, London, a CERN Tier-2 institute.

He received his MSc. in Computer and Electronic Security from Queen’s University, Belfast.

In 2010, he joined the Knowledge and Data Engineering research group at Queen’s, where he obtained a Ph.D. for his research into algorithms to discover patterns and anomalies in graphs. He has also worked as an extra in Game of Thrones seasons 2, 3 and 4.

Michael is looking forward to joining the IT Department (Storage Group) in spring 2017.

Mentors

Sebastien Ponce, Ivica Puljak

Lectures

Algorithms for Anomaly Detection

The concept of statistical anomalies, or outliers, has fascinated experimentalists since the earliest attempts to interpret data. We want to know why some data points don’t seem to belong with the others: perhaps we want to eliminate spurious or unrepresentative data from our model. Or, the anomalies themselves may be what we are interested in: an outlier could represent the symptom of a disease, an attack on a computer network, a scientific discovery, or even an unfaithful partner.

We start with some general considerations, such as the relationship between clustering and anomaly detection, the choice between supervised and unsupervised methods, and the difference between global and local anomalies. Then we will survey the most representative anomaly detection algorithms, highlighting what kind of data each approach is best suited to, and discussing their limitations. We will finish with a discussion of the difficulties of anomaly detection in high-dimensional data and some new directions for anomaly detection research.
Daniel LANZA

CERN, Switzerland

At CERN, an organisation that Daniel joined more than 2 years ago, he is working on developing and providing Big Data solutions.

During his two Degrees and two Masters, where he studied computer science, telecommunications and Big Data, he has been specially interested in Evolutionary Computation, field of knowledge where he has several publications.

His attention has been mainly in joining the two fields he loves, Big Data and Evolutionary Computation, hence he has worked on integrating them to speed up evolutionary processes by running them on distributed environments like Hadoop. However, he has also worked in a bloat control mechanism for genetic programming algorithms to avoid the grow in size and computational cost of individuals (solutions).

Mentors
Nikos Kasioumis, Ivica Puljak

Lectures
Applying natural evolution for solving computational problems

Darwin’s natural evolution theory has inspired computer scientists for solving computational problems. In a similar way to how humans and animals have evolved along millions of years, computational problems can be solved by evolving a population of solutions through generations until a good solution is found.

In the first lecture, the fundaments of evolutionary computing (EC) will be described, covering the different phases that the evolutionary process implies. ECJ, a framework for researching in such field, will be also explained.

In the second lecture, genetic programming (GP) will be covered. GP is a sub-field of EC where solutions are actual computational programs represented by trees. Bloat control and distributed evaluation will be introduced.
Lorena LOBATO PARDAVILA

_CERN, Switzerland and Universidad de Oviedo, Spain_

Lorena works as a DevOps Engineer in the Computing and Monitoring group of the IT department, having worked previously on Oracle replication technologies within the Data Bases group and as a software developer for the GLIB project in the Electronic Systems group in the PH department.

After receiving her MSc in Computer Science at the University of Vigo, she subsequently worked for different companies as a System Analyst and started working for CERN in 2011. She is a positive-thinker, and an enthusiastic and dynamic person with strong interests in medical applications and cognitive systems, technological development and engineering systems, and how they can be applied to solving IT challenges. She is also a passionate about music, travels, literature and sport in general.

**Mentors**

Alberto Pace, Sebastien Ponce

**Lecture**

Virtual Machine Images Management in Cloud Environments

Nowadays, the demand for scalability in distributed systems has led a design philosophy in which virtual resources need to be configured in a flexible way to provide services to a large number of users. The configuration and management of such an architecture is challenging (e.g.: 100,000 compute cores on the private cloud together with thousands of cores on external cloud resources). There is the need to process CPU intensive work whilst ensuring that the resources are shared fairly between different users of the system, and guarantee that all nodes are up to date with new images containing the latest software configurations. Different types of automated systems can be used to facilitate the orchestration.

CERN’s current system, composed of different technologies such as OpenStack, Packer, Puppet, Rundeck and Docker will be introduced and explained, together with the process used to create new Virtual Machines images at CERN.
Eamonn MAGUIRE

Pictet Asset Management, Switzerland

Eamonn completed his DPhil (PhD) at the University of Oxford in computer science, focused on data visualization, in particular the systematisation of glyph design. His research interests are in the merging of machine learning and visual analytics, where he currently plies his trade as a Data Scientist at Pictet Asset Management in Geneva.

Until November 2016 he was a Senior Marie Curie COFUND Fellow at CERN where he led development of the new hepdata.net platform and contributed to numerous other visualization and data projects at CERN. Before that, he was the lead software engineer at the Oxford University e-Research Centre, where he led development of bioinformatics tools and a visual analytics platform for corporate insider threat detection.

Mentors
Olivier Couet, Danilo Piparo

Lectures
Creating Effective Data Visualizations

In this course, I aim to give an overview of data visualisation as a field, including many of the important theoretical groundings in data visualization. We will explore the different ways of representing visual information, and the strengths/weaknesses of those approaches.

Using real-world case studies, I will demonstrate techniques and best practices for visualizing complex multi-dimensional data common to high energy physics and other fields.
## Monday 6 March

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:00 - 14:15</td>
<td>A word from the IT Department Head</td>
<td>Frédéric HEMMER</td>
</tr>
<tr>
<td>14:15 - 14:30</td>
<td>Introduction to the inverted CSC</td>
<td>Sebastian LOPJENSKI</td>
</tr>
<tr>
<td>14:30 - 15:30</td>
<td>Let your machine do the learning - Lecture 1</td>
<td>Daniel CAMPORA PEREZ</td>
</tr>
<tr>
<td>15:30 - 16:00</td>
<td>Coffee</td>
<td></td>
</tr>
<tr>
<td>16:00 - 17:00</td>
<td>Algorithms for Anomaly Detection - Lecture 1</td>
<td>Michael DAVIS</td>
</tr>
</tbody>
</table>

## Tuesday 7 March

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 - 10:00</td>
<td>Applying natural evolution for solving computational problems - Lecture 1</td>
<td>Daniel LANZA</td>
</tr>
<tr>
<td>10:00 - 11:00</td>
<td>Distributed consensus and fault tolerance - Lecture 1</td>
<td>Georgios BITZES</td>
</tr>
<tr>
<td>11:00 - 11:30</td>
<td>Coffee</td>
<td></td>
</tr>
<tr>
<td>11:30 - 12:30</td>
<td>Algorithms for Anomaly Detection - Lecture 2</td>
<td>Michael DAVIS</td>
</tr>
<tr>
<td>12:30 - 14:00</td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>14:00 - 15:00</td>
<td>Virtual Machine Images Management in Cloud Environments</td>
<td>Lorena LOBATO PARDAVILA</td>
</tr>
<tr>
<td>15:00 - 15:30</td>
<td>Coffee</td>
<td></td>
</tr>
<tr>
<td>15:30 - 16:30</td>
<td>Let your machine do the learning - Lecture 2</td>
<td>Daniel CAMPORA PEREZ</td>
</tr>
<tr>
<td>Time</td>
<td>Session</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>09:00 - 10:00</td>
<td>L8</td>
<td>Creating Effective Data Visualizations - Lecture 1</td>
</tr>
<tr>
<td>10:00 - 11:00</td>
<td>L9</td>
<td>Distributed consensus and fault tolerance - Lecture 2</td>
</tr>
<tr>
<td>11:00 - 11:30</td>
<td></td>
<td>Coffee</td>
</tr>
<tr>
<td>11:30 - 12:30</td>
<td>L10</td>
<td>Applying natural evolution for solving computational problems - Lecture 2</td>
</tr>
<tr>
<td>12:30 - 14:00</td>
<td></td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00 - 15:00</td>
<td>L11</td>
<td>Creating Effective Data Visualizations - Lecture 2</td>
</tr>
<tr>
<td>15:00 - 15:30</td>
<td></td>
<td>Coffee</td>
</tr>
<tr>
<td>15:30 - 16:30</td>
<td>L12</td>
<td>Let your machine do the learning - hands-on session</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:30 - 16:45</td>
<td></td>
<td>Closing remarks</td>
</tr>
</tbody>
</table>
Introduction

What is learning?

Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed (A. Samuel, 1959).

Any change in a system that allows it to perform better the second time on repetition of the same task (H. Simon).
Find the (conceptual) difference

Supervised
Unsupervised

Learning types and paradigms

- Learning types
  - Supervised
  - Unsupervised
  - Reinforcement
- Learning paradigms
  - Learning by memorizing
  - Clustering
  - Inductive learning
  - Analogy-based learning
  - Discovery
  - Genetic algorithms, neural networks
Supervised learning

Diving in with an example

Let’s start with a supervised learning problem. The following dataset contains the living area and prices for housing in Portland.

<table>
<thead>
<tr>
<th>Living area (feet²)</th>
<th>Price (1000$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>400</td>
</tr>
<tr>
<td>1600</td>
<td>330</td>
</tr>
<tr>
<td>2400</td>
<td>369</td>
</tr>
<tr>
<td>1416</td>
<td>232</td>
</tr>
<tr>
<td>3000</td>
<td>540</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Daniel Hugo Cámara Pérez

Let your machine do the learning

Supervised learning

Notation

We will need some notation,

- **x** Input variable
- **y** Output variable
- \((x_\mu, y_\mu); \mu = 1, \ldots, m\) Training set
- \(X, Y\) Space of input and output variables

In our example,

- **x** Living area
- **y** Price
- \((x_\mu, y_\mu); \mu = 1, \ldots, m\) Housing dataset
- \(X = Y = \mathbb{R}\)

Daniel Hugo Cámara Pérez

Let your machine do the learning

Supervised learning

Diving in with an example (2)

Let’s plot the data.

Given this data, how can we predict the prices of other houses in Portland as a function of their area?

What do we want?

Given a training set, we want to learn a function \(h : X \rightarrow Y\) so that \(h(x)\) predicts as certainly as possible the output \(y\). Historically, this is called a hypothesis.

For our housing example, we want to learn a function that predicts the pricing of the house, based on the area of it. We will be using our training set to generate this function.

We can later validate our data by using a separate test set. The training and test sets are disjoint.
Supervised learning

Classification and regression

The problems where the target variable $y$ is continuous, such as our housing one, are called regression problems. In opposition, when the target value can only take a few values, we call it a classification problem.

Binary classification problems are very common, and historically 1 and 0 are also called the positive and negative class respectively, and are denoted by $+$ and $-$.

![Figure 1: Multiclass classification application: LHCb Particle Identification.](http://indico.cern.ch/event/589985)

Linear Regression example

Now we have the notation right, we can make our example a bit more interesting by extending the dataset with a new feature. From now on, we will work this example out in detail.

<table>
<thead>
<tr>
<th>Living area ($\text{feet}^2$)</th>
<th>Bedrooms</th>
<th>Price ($\text{1000$s}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>3</td>
<td>400</td>
</tr>
<tr>
<td>1600</td>
<td>3</td>
<td>330</td>
</tr>
<tr>
<td>2400</td>
<td>3</td>
<td>369</td>
</tr>
<tr>
<td>1416</td>
<td>3</td>
<td>232</td>
</tr>
<tr>
<td>3000</td>
<td>4</td>
<td>540</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

As an initial choice, we can approximate $y$ as a function of $x$:

\[ h_\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \] (1)

The $\theta_i$ are the weights parameterizing the space of linear functions mapping from $X$ to $Y$. For simplicity and assuming $x_0 = 1$, we can also write the above as:

\[ h(x) = \sum_{j=0}^{n} \theta_j x_j = (\theta_0 \theta_1 \theta_2 \ldots) \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \end{pmatrix} \] (2)

Model

As an initial choice, we can approximate $y$ as a function of $x$: $h_\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ (1)

The $\theta_i$ are the weights parameterizing the space of linear functions mapping from $X$ to $Y$. For simplicity and assuming $x_0 = 1$, we can also write the above as $h(x) = \sum_{j=0}^{n} \theta_j x_j = (\theta_0 \theta_1 \theta_2 \ldots) \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \end{pmatrix}$ (2)
How do we learn \( \theta \)?

Since what we want is to make \( h(x) \) as close as possible to \( y \), we can define a function that measures how close it is. Thus, we define the cost function:

\[
E(\theta) = \frac{1}{2} \sum_{\mu=1}^{m} (h_\theta(x_\mu) - y_\mu)^2
\]  

(3)

Perhaps you recognize this function. It is the least-squares cost function that gives rise to the ordinary least squares regression model.

Note we use the subscript \( \mu \) to iterate over the training example, and the subscript \( j \) to iterate over the variables in our hypothesis (like in eq. 2).

The update rule

For a single training example \( (x_\mu, y_\mu) \), this gives the following update rule:

\[
\forall j \in 0, \ldots, n \quad \theta_j := \theta_j + \alpha (y_\mu - h_\theta(x_\mu))x_{\mu j}
\]  

(5)

This rule is called the Least Mean Squares update rule. Note that the subscripts \( \mu \) refer to a specific example from the training set. If we want to generalize to the whole training set, there are two ways of doing so. This is one of them,

\[
\forall j \in 0, \ldots, n \quad \theta_j := \theta_j + \alpha \sum_{\mu=1}^{m} (y_\mu - h_\theta(x_\mu))x_{\mu j}
\]  

(6)

This is called the batch gradient descent method.

The gradient descent algorithm

We want to choose a \( \theta \) that will minimize the cost function \( E(\theta) \). Let’s consider the gradient descent algorithm, which starts with some initial \( \theta \), and performs the update:

\[
\forall j \in 0, \ldots, n \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} E(\theta)
\]  

(4)

This function performs an update on all \( \theta_j \) terms. \( \alpha \) is called the learning rate, and it will determine how fast will \( \theta \) change. This will impact how our algorithm learns, as we will see later.
Supervised learning

No more equations for now

Now we are ready to take over our supervised learning problem.

![Image](Pinky and the Brain: Take Over the World)

Our examples solved, at last

Here is a plot of $h_{\theta}(x)$ as a function of $x$ (area), along with the training set,

![Image](Plot of $h_{\theta}(x)$)

If we include the number of bedrooms, we get $\theta = \{89.60, 0.14, -8.74\}$.

Gradient descent extended

Coming back to equation 5, instead of updating $\theta$ based on the whole training set at every step (like on eq. 6), we could update the parameters according to a single training example at a time. That would yield the following equation:

$$\forall \mu \in 1, \ldots, m \quad \forall j \in 0, \ldots, n \quad \theta_j := \theta_j + \alpha(y_\mu - h_\theta(x_\mu))x_{\mu j} \quad (7)$$

This algorithm is called the stochastic gradient descent, and it performs much faster than the batch gradient descent. However, as a consequence, it may oscillate when getting closer to the minimum value.
Beware of local minima

In our little examples, the function we are trying to minimize is convex (it has a global minimum), so our gradient descent model is guaranteed to find it, with a varying performance dependent on the learning rate \( \alpha \) and the specifics of the method used.

However, this is not always the case: In cases where local minima exist, chances are we are not getting the good one.

Overfitting

In some cases, if the training set is not well balanced, your model may learn an overfit solution to your problem.

Nature has the best ideas
Making a model out of it

The signal output of a neuron can either cause excitation or inhibition in the neuron it is connected to. When a neuron sends an excitatory signal to another neuron, then this signal will be added to all of the other inputs of that neuron. If it exceeds a given threshold then it will cause the target neuron to fire an action potential, if it is below the threshold then no action potential occurs.

Artwork by Synaptidude at en.wikipedia.

Historical highlights

Among the first attempts at a model there is the McCulloch and Pitts model (1943, [4]). Neurons have a binary state (firing or not firing), and its state depends on the neighbouring neurons.

In 1958, Rosenblatt et al. [5] worked on a simple model called Perceptron which consisted in organising neurons in two layers. Widrow and Hoff developed a similar concept called the adaline [6]. These models were very successful because weights could be tuned by a learning algorithm.

However, Marvin Minsky, in 1967 [7], proved that simple perceptrons could not address non-linearly separable problems.

And so the field was sort of dead for a couple of decades.
Historical highlights (4)

Rumelhart et al. in 1986, developed the Backpropagation algorithm [8] which allowed to train multilayer perceptrons and so broke the non-linear separability curse. That caused a boom in neural network applications.

In the mid 90s, Independent Component Analysis was developed using neural networks. The use of the expression Deep Learning in the context of Artificial Neural Networks was introduced by Aizenberg et al. in 2000 [9].

Deep learning has revolutionized several major fields.

Figure 3: Lee Sedol vs AlphaGo (https://gogameguru.com/tag/deepmind-alphago-lee-sedol/)

The game of Go is an ancient board game, played by placing white and black stones on one of the 19x19 intersections on a board. Formally, the problem lies in the EXP computational complexity class, so computer programs tend to be weak against Go professionals.

In 2016, a best-of-5 match was held between Lee Sedol, one of the best players in the world, and AlphaGo, a program using deep neural networks and tree search [10]. AlphaGo was trained over a large database of games, and used a distributed system of 1920 CPUs and 280 GPUs in the backend. AlphaGo won the match 4-1.

The perceptron

You can think of a perceptron as a mathematical model for a single neuron.

It consists of:
- Inputs $-1, x_1, x_2, \ldots, x_n$
- Weights $w_0, w_1, w_2, \ldots, w_n$
- Activation function $f(x)$

With these ingredients, the output of the perceptron is calculated as follows,

$$y = f \left( \sum_{j=0}^{n} w_j x_j \right)$$ (8)
Activation functions

The activation function \( f(x) \) defines the relation between the states of the neighbouring neurons and the state of the neuron under study. It must be chosen according to the nature of the state of the neuron.

Here are some typical activation functions:

- \( f(x) = \begin{cases} 
1 & \text{if } x > 0 \\
-1 & \text{if } x \leq 0
\end{cases} \)
- \( f(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{if } x \leq 0
\end{cases} \)
- \( f(x) = \frac{1}{1 + e^{-x}} \)

The sigmoid function

In particular, the activation function \( f(x) = \frac{1}{1 + e^{-x}} \) is called the sigmoid function. It has the particularity that it tends towards 1 as \( x \to \infty \) and towards 0 as \( x \to -\infty \). Moreover, it’s always bounded between 0 and 1.

\[ f(x) = \frac{1}{1 + e^{-x}} \]

The sigmoid function (2)

As we will see on the next class, it’s useful to have an activation function that is differentiable to let our network learn with the backpropagation algorithm. The sigmoid function is indeed differentiable,

\[ f'(x) = f(x)(1 - f(x)) \] (9)

which is kind of neat.

An example

Here’s the AND gate state table:

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A perceptron can simulate this with the following configuration:

- Weights \( w = \{1.5, 1, 1\} \)
- Activation function \( f(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{if } x \leq 0
\end{cases} \)
More examples

Other basic gates, like OR, NOT, NAND and NOR, are also computable in this fashion.

However, is the XOR gate computable with a single perceptron?

Linear separability

There is an inherent limitation in terms of expressive power of the perceptron. The generic activation function,

\[ y = f(\sum_{j=0}^{n} w_j x_j) = f(\vec{w} \cdot \vec{x}) \] (10)

separates input hyperspace by the hyperplane of the equation,

\[ \vec{w} \cdot \vec{x} = 0 \] (11)

and so, only linearly separable functions can be expressed by a perceptron.

Gradient descent revisited

We can apply the gradient descent technique we have already seen for learning the weights of the perceptron. With no further ado, the cost function given by the quadratic error for a single training example \((x_{\mu}, y_{\mu})\) is,

\[ E(\vec{w}) = \frac{1}{2}(f(input_{\mu}) - y_{\mu})^2 \] (12)

The generic gradient descent can be then formulated as,

\[ \forall j \in 1,...,n \quad w_j := w_j + \alpha \sum_{\mu=1}^{m} (y_{\mu} - f(input_{\mu})) f'(input_{\mu}) x_{\mu j} \] (13)

Gradient descent revisited (2)

Considering we have a training set consisting of \(m\) examples, we can either learn from all of them at the same time, or apply one after the other. As we saw before, we can then formulate **batch gradient descent**:

\[ \forall j \in 1,...,n \quad w_j := w_j + \alpha \sum_{\mu=1}^{m} (y_{\mu} - f(input_{\mu})) f'(input_{\mu}) x_{\mu j} \] (14)

And **stochastic gradient descent**:

\[ \forall \mu \in 1,...,m \quad \forall j \in 1,...,n \quad w_j := w_j + \alpha (y_{\mu} - f(input_{\mu})) f'(input_{\mu}) x_{\mu j} \] (15)
Neural networks foundations

Gradient descent revisited (3)

When considering learning from a set of $m$ examples, the error we want is an average of the errors across all examples from the set. Thus, we can formulate the mean quadratic error:

$$E(w) = \frac{1}{m} \sum_{\mu=1}^{m} \frac{1}{2}(f(input_{\mu}) - y_{\mu})^2$$ (16)

In a nutshell

We have reviewed the concepts of supervised learning and the mathematical model of the perceptron. For both, we have defined a cost function to optimise, and derived the gradient descent methods for learning the parameters and weights, respectively.

A single perceptron, however, is not enough to approximate any continuous function. In the next course we will review how to overcome the linear separability, and use neural networks for various real applications.

Bibliography

Resources I


Let your machine do the learning

Neural Networks

Daniel Hugo Cámara Pérez

Universidad de Sevilla
CERN

Inverted CERN School of Computing, 6th - 8th March, 2017

Outline

1 Introduction
2 Neural networks
3 Convolutional neural networks
4 Where to go from here
5 Bibliography

Intro

Section 1

Introduction

In the last episode

A perceptron consists of:

- Inputs $-1, x_1, x_2, \ldots, x_n$
- Weights $w_0, w_1, w_2, \ldots, w_n$
- Activation function $f(x)$

With these ingredients, the output of the perceptron is calculated as follows,

$$y = f\left(\sum_{j=0}^{n} w_j x_j\right)$$ (1)
In the last episode (2)

We can apply the gradient descent technique we have already seen for learning the weights of the perceptron. The cost function given by the quadratic error for a single training example \((x_\mu, y_\mu)\) is,

\[
E(\vec{w}) = \frac{1}{2}(f(input_\mu) - y_\mu)^2
\]  

(2)

The generic gradient descent can be then formulated as,

\[
\forall j \in 1, \ldots, n \quad w_j := w_j + \alpha(y_\mu - f(input_\mu))f'(input_\mu)x_{\mu j}
\]  

(3)

Connecting the dots

In our previous lecture, we extensively covered the case study for a single perceptron, and found they can only express linearly separable functions, which does not make them particularly interesting. However, we can overcome this limitation by connecting several perceptrons into a neural network.

Network topology

In a network, the topology defines which neurons are connected to which others. Each connection is calibrated with an appropriate weight. In general connections need not be symmetric.

Among the most common topological structures, we find the layer. Neurons on the same layer do not have any interaction with each other. Instead, they interact with other layers.
Feedforward networks

Feedforward networks have their neurons organised in layers. The input layer receives external data, and the output layer produces the result. In between, there may be any number of hidden layers. The information moves in only one direction (forward) from the input layers, through the hidden layers, and until the output layers. There are no cycles on these networks.

Multilayer perceptron

Now that we consider a network consisting of a number of perceptrons, we can generalize our definition for a particular topology. Let’s consider a multilayer perceptron as the building block of our feedforward network.

Other network topologies

Be aware that many other topologies exist, although we are not going to discuss them here. Suffice it to say that, upon the problem under study, the choice of the topology is an important one.

Cybenko (1989 [5])
Multilayer perceptrons with one hidden unit are universal approximators of continuous functions (or with a finite number of discontinuities).
Gradient descent

In our last course, we covered the gradient descent method. We used it to learn the $\theta$ terms for linear regressions:

$$\forall j \in 0,...,n \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} E(\theta) \quad (4)$$

We also used it for learning the weights of a single perceptron:

$$\forall j \in 1,...,n \quad w_j := w_j + \alpha (y_\mu - f(input_\mu)) f'(input_\mu) x_j \quad (5)$$

Backpropagation

The algorithm of backpropagation consists in applying gradient descent techniques on the quadratic error for perceptrons. This algorithm revolutionized the machine learning field in the mid-80s, as it allowed a neural network of an arbitrary complexity to be effectively trained.

Here, we will focus on multilayer perceptrons, and derive the backpropagation algorithm for them. Adjust your seat belts.

Backpropagation (2)

If $x_i^{[c]}$ denotes the state of neuron $i$ on layer $c$, then:

$$x_i^{[c]} = f\left(\sum_{j=0}^{n} w_{ij}^{[c-1]} x_j^{[c-1]}\right) \quad (6)$$

It is worth taking some time to get the notation right, as we start populating indices up and down. The superscript $c$ refers to the layers. In each layer, we have an arbitrary number of perceptrons, denoted by subscript $i$. Lastly, the subscript $j$ refers to the inputs and weights of perceptron $i$.

$x_i^{[1]} = x_i$ corresponds to the input layer, and $x_i^{[N]} = y_i$ to the output layer.

Backpropagation (3)

When the input is fed with training pattern $(x_\mu, y_\mu)$, the neuron states take the corresponding values:

$$x_i^{[c]_\mu}, \quad x_i^{[0]_\mu} = x_i, \quad x_i^{[N]_\mu} = y_i \quad (7)$$

Just as we did for the simple perceptron we define the quadratic error and propose the gradient descent optimisation technique,

$$E(\vec{w}) = \frac{1}{2} \sum_{\mu=1}^{N} \sum_{i=0}^{n} (x_i^{[c]_\mu} - y_{\mu i})^2 \quad (8)$$

$$\frac{\partial E}{\partial w_{ij}^{[c]}} = -\sum_{\mu=1}^{N} \sum_{j=0}^{n} (y_{\mu i} - x_i^{[c]_\mu}) \frac{\partial x_j^{[c]}}{\partial w_{ij}^{[c]}} \quad (9)$$
Neural networks

**Backpropagation (4)**

We will not cover the development of this equation in depth. Suffice it to say that for the last layer we have the same formula as we had for the simple perceptron. Therefore, the backpropagation learning rule is defined as follows:

\[
\forall i, j \quad w_{ij}^{[c]} := w_{ij}^{[c]} + \alpha \sum_{\mu=1}^{N} \delta_{ij}^{[c+1]} x_{\mu j}^{[c]} \quad (10)
\]

where,

\[
\delta_{ij}^{[N]} = (y_{ij} - x_{ij}^{[N]}) f'(\sum_{s=0}^{N-1} w_{js}^{[N-1]} x_{is}^{[N-1]})
\]

\[
\delta_{ij}^{[k]} = \sum_{r=0}^{N} \delta_{ij}^{[k+1]} w_{js}^{[k]} f'(\sum_{t=0}^{k-1} w_{st}^{[k-1]} x_{it}^{[k-1]})
\]

**Remarks about backpropagation**

- Modification of each weight depends only on objects living on the neurons connected by its weight.
- Weight variations are larger for the last layers and very small for inner layers. Large networks are very hard to train.
- Learning stops by
  - fixed number of epochs
  - overtraining
  - target error achieved
- Weights are usually randomly initialised, with small values.

**Backpropagation (5)**

Here is some visual aid to help you through the formulae:

(a) Detail of layer notation.

(b) Detail of neuron notation.

**A visual example**

Figure 2: Tensorflow has some pretty examples to visualize the learning process. (http://playground.tensorflow.org)
Convolutional neural networks

Section 3

What is a convolution?

An image convolution is a transformation pixel by pixel, done by applying to an image some transformation defined by a set of weights, also known as a filter. Let $s$ be a set of source pixels, and $w$ a set of weights, a pixel $y$ is transformed as follows:

$$y = \sum_{i=0}^{n} s_i w_i$$

Examples of filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>$\begin{bmatrix} 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>Edge detection</td>
<td>$\begin{bmatrix} -1 &amp; -1 &amp; -1 \ -1 &amp; 8 &amp; -1 \ -1 &amp; -1 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Sharpen</td>
<td>$\begin{bmatrix} 0 &amp; -1 &amp; 0 \ -1 &amp; 5 &amp; -1 \ 0 &amp; -1 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>Blur</td>
<td>$\begin{bmatrix} 1/9 &amp; 1/9 &amp; 1/9 \ 1/9 &amp; 1/9 &amp; 1/9 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

Dealing with images

Convolutional neural networks are very similar to ordinary neural networks: They are made up of neurons with learnable weights. However, convolutional neural networks make the explicit assumption that the inputs are images.

This fact allows to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.
Regular neural networks don’t scale well to full images. To see this, let’s have a look at a subset of the CIFAR-10 dataset:

This dataset is composed of images of size 32x32x3 (width x height x color channels), categorized to what they contain. Note even though these are 2D images, due to the color channels, we are dealing with volumes.

Convolutional layers

A convolutional layer consists of a set of learnable filters. Every filter is spatially small in width and height, but extends through the full depth of the input volume.

During the forward pass, we slide each filter across a local region of the width and height of the input volume, and compute dot products between the entries of the filter and the input at any position, as in eq. 11. As we slide the filter over the width and height of the input volume, we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position.

For instance, given an input volume of sizes 32x32x3 (like CIFAR-10), we could have a filter of size 5x5 and slide it across the input. This would result in a total of 5 * 5 * 3 = 75 input weights. Here is an example (http://cs231n.github.io/assets/conv-demo/index.html).
Convolutional layers spatial arrangement

Three parameters control the size of the output volume:

- **Depth**: Number of filters we would like to use, each searching for a different characteristic.
- **Stride**: Determines how the filter is slid across the input. It is uncommon to slide more than three pixels at a time.
- **Zero-padding**: Determines the number of pixels filled with zeroes around the border. Sometimes, this is useful to allow certain strides.

We can compute the spatial size of the output volume as a function of the input volume size $W$, the filter size $F$, stride $S$ and padding $P$ as:

$$W - F + 2P + 1$$  \hspace{1cm} (12)

Convolutional layers parameter sharing

Following the assumption that if one feature is useful to compute at some spatial position $(x_0, y_0)$, then it should also be useful to compute at a different position $(x_1, y_1)$, it is possible to reduce dramatically the number of parameters (weights) in a CNN.

![Example filters learnt by Krizhevsky et al. [6], who won the ImageNet challenge (http://www.image-net.org/challenges/LSVRC in 2012. The first convolutional layer had a size of 55\times55\times96 = 290 400 neurons with 11\times11\times3+1 = 364 inputs each. This makes for 105 705 600 weights, only feasible due to parameter sharing.](image)

Pooling layers

The function of a pooling layer, also known as a subsampling layer, is to progressively reduce the spatial size of the representation, to reduce the amount of parameters and computation in the network. The pooling is typically done using the average or maximum function, applied to the subset in consideration.

![Pooling using the maximum function, with 2x2 filters and stride 2.](image)

Other types of layers

- **Normalization layers**: Many types of normalization layers have been proposed, for instance with the intention of implementing inhibition schemes observed in the biological brain. However, these layers have since fallen out of favor because in practice their contribution has been shown to be minimal.
- **Fully-Connected layers**: Neurons in a fully-connected layer have full connections to all activations in the previous layer, as seen in regular neural networks. They are used in the output layers of CNNs.
Example: Classifying birds

Convolutional neural networks

Well, let’s actually do it

For this example, we will use tensorflow(https://www.tensorflow.org/), an open source library for numerical computation using data flow graphs. It’s quite powerful, and has us covered for what we need.

Our input will be the CIFAR-10 data set, combined with the Caltech-UCSD Birds-200-2011 data set. This way we have 18,000 pictures of birds, and 52,000 pictures of not birds.

Figure 6: Birds come in many colors and shapes. Notice some pictures are actually cropped, or not oriented the same way, and color conditions are very different.

Convolutional neural networks

Our convolutional neural network

Our convolutional neural network will look like this,

- Input is 32x32x3
- Convolution layer of size 32x32x3
- Max pooling with stride 2
- Convolution layer of size 64x64x3
- Convolution layer of size 64x64x3
- Max pooling with stride 2
- Fully-Connected network of 512 neurons
- Dropout some data randomly to prevent overfitting
- Fully-Connected network with two outputs (0: no-bird, 1: bird)

Prototyping in tensorflow

```python
# Input is a 32x32 image with 3 color channels
network = input_data(shape=[None, 32, 32, 3],
data_preprocessing=img_prep,
data_augmentation=img_aug)

# Network layers
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 64, 3, activation='relu')
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 512, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 2, activation='softmax')
```
Convolutional neural networks

Prototyping in tensorflow (2)

```python
network = regression(network, optimizer='adam',
                     loss='categorical_crossentropy',
                     learning_rate=0.001)

model = tflearn.DNN(network,
                     checkpoint_path='bird-classifier.tfl.ckpt')

# Train the network for 100 epochs
model.fit(X, Y, n_epoch=100,
          shuffle=True, validation_set=(X_test, Y_test),
          show_metric=True, batch_size=96,
          snapshot_epoch=True,
          run_id='bird-classifier')
```

Source code (https://gist.github.com/ageitgey/a2a03154bda7f6f86da4e34bf52bebf)

Daniel Hugo Campora Perez
Let your machine do the learning
iCSC 2017 37 / 47

Checking the efficiency of our network

Finally, let’s check how we did. Depending on how we separated our data set into a training set and a validation set, results may vary.

Running the above code reports a 95% efficiency in recognizing whether we found a bird or not. We can also have a look at the precision and recall:

\[
\text{precision} = \frac{\text{true positives}}{\text{all positive guesses}}
\]
\[
\text{recall} = \frac{\text{true positives}}{\text{total birds in dataset}}
\]

We obtain a 97.11% of precision, and a 90.83% recall. Not so bad!

Daniel Hugo Campora Perez
Let your machine do the learning
iCSC 2017 38 / 47

Where to go from here

Section 4

Paradigms

Now that you have strong foundations of neural networks, if you want to keep digging there is a lot more you can explore. Some of the more interesting paradigms out there are:

- Recurrent neural networks [7]
- Deep belief networks [8]

You can also find a plethora of well-documented applications. The following are a few of the most recent results neural networks have produced in the last years.

Daniel Hugo Campora Perez
Let your machine do the learning
iCSC 2017 39 / 47
Speech recognition problems have successfully been solved with recurrent neural networks [9], [10].

Here are some materials on self driving vehicles [11] as well as traffic signal recognition [12].

ATLAS Higgs ML challenge (http://opendata.cern.ch/collection/ATLAS-Higgs-Challenge-2014) was held in 2014. The winner, G. Melis, used a deep neural network to achieve his result. Find his presentation here (https://indico.cern.ch/e/382895).


Astronomers prefer to reject completely observations which they judge to be too wide of the truth, while retaining the rest… I see no way of drawing a dividing line between those that are to be utterly rejected and those that are to be wholly retained; it may even happen that the rejected observation is the one that would have supplied the best correction to the others… I think each and every observation should be admitted whatever its quality…

— Daniel Bernoulli

The most probable choice between several discrepant observations and the formation therefrom of the most likely induction (1777)
In August 1944, Mr. Hadlum left home on a period of military service. 349 days later (12 Aug 1945), Mrs. Hadlum gave birth. On his return, Mr. Hadlum filed for divorce.

Average human gestation period is 280 days. Mr. Hadlum judged that 349 days was an outlier. The question is what process gave rise to it:

- An unusually long gestation?
- Adultery (asserted by Mr. Hadlum)
Introuduction

Global Anomalies
Local Anomalies
High-Dimensional Data
Other Approaches

What is an Anomaly?
Things to Consider
Overview

Hadlum vs. Hadlum (1949)

- Average human gestation period is 280 days
- Mr. Hadlum judged that 349 days was an outlier
- The court ruled that the observation was valid, if extreme
- Mr Hadlum claimed that the observation was a contaminant
  ▶ He did not want the observation to be rejected
  ▶ He wanted it to be identified, with appropriate consequences

What is an Anomaly?

- Mr. Hadlum judged that 349 days was an outlier
- The court ruled that the observation was valid, if extreme
- Mr Hadlum claimed that the observation was a contaminant
  ▶ He did not want the observation to be rejected
  ▶ He wanted it to be identified, with appropriate consequences

Definition of an Outlier

- An outlying observation, or ‘outlier’, is one that appears to deviate markedly from other members of the sample in which it occurs.

—F.E. Grubbs

Procedures for detecting outlying observations in samples (1969)

We shall define an outlier in a set of data to be an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.

—V. Barnett and T. Lewis

Outliers in Statistical Data (1978)

Things to Consider

How to Find Anomalies

Naïve Approach

- Define a region representing normal behaviour. Declare anything outside that region as an anomaly.

This is difficult because:

- Defining the normal region is difficult
- Normal behaviour may not be static
- What is considered normal/anomalous may depend on the domain
What is an Anomaly?

- Define a region representing normal behaviour. Declare anything outside that region as an anomaly.

This is difficult because:

- Defining the normal region is difficult
- Normal behaviour may not be static
- What is considered normal/anomalous may depend on the domain

Naïve Approach

Data instance is usually a vector
Data can have different characteristics:

- Univariate
- Multivariate
- Attribute type can be binary/categorical/ordinal/continuous
- Multivariate data can have attributes of more than one type
- Structured data: temporal, spatial, sequence, graph
Things to Consider
Supervised/Semi-supervised/Unsupervised Scenarios

Supervised Approaches
- Requires training data for normal and anomaly classes
- Build a classifier (predictive model) for all classes
- Unseen data is compared to the model and classified as normal or anomalous
- Usually, anomalous instances are far fewer than normal instances
- Difficult to obtain accurate class labels for the anomaly instances

Semi-supervised Approaches
- Requires training data for the normal class only
- Build a one-class classifier
- Unseen data is compared to the model and classified as belonging to the normal class, or not
- Difficult to ensure that training data does not contain any anomalies

Unsupervised Approaches
- Assumption
  - Normal instances are much more common than anomalous instances
- Does not require any training data
- Difficult to determine the threshold between normal data and outliers

Labelling
- Binary output
- Data objects are labeled either as normal or outlier
Intr oduction

Global Anomalies

Local Anomalies

High-Dimensional Data

Other Approaches

Things to Consider

Labelling vs. Scoring

Scoring

- Continuous output, representing the degree of “outlierness”
- For each object an outlier score is computed
- Data objects can be sorted and ranked according to their scores
- Many scoring approaches focus on determining the top-n outliers
- Convert scores to labels by thresholding on the score

Global Outliers

Assumption

There is only one mechanism that generates the normal data instances

- Reference set is all other data objects
- Other outliers are also in the reference set and may distort the results

Overview

- Introduction
- Global Anomaly Detection
  - Statistical Approaches
  - Classification-based Approaches
  - Clustering-based Approaches
- Local Anomaly Detection
  - Distance-based Approaches
  - Density-based Approaches
- Anomaly Detection in High-dimensional Data
- Other Approaches

Local Outliers

- No assumption on the number of normal mechanisms
- Reference set is a small subset of data objects
- Resolution of the reference set can vary from a single object (local) to the entire database (global)
- Main problem: how to choose the optimal reference set
Global Anomaly Detection

- Statistical Approaches
  - Parametric
  - Nonparametric
- Classification-based Approaches
- Clustering-based Approaches

### Statistical Approaches

#### Parametric Methods

- Normal observations $x$ are generated by a process which is modelled as a parametric distribution
- Model is a Probability Density Function (PDF) $f(x, \theta)$
- Parameters $\theta$ are estimated from training data
- Labelling: Statistical hypothesis test. Null hypothesis $H_0$ is that $x$ was generated using the estimated distribution $f(x, \theta)$. If the statistical test rejects $H_0$, $x$ is an anomaly.
- Scoring: Anomaly score for instance $x \in x$ is the inverse of the PDF $f(x, \theta)$

#### Assumption

Normal data instances occur in high-probability regions of a stochastic model; anomalies occur in the low-probability regions of the model.

- Training: Fit a statistical model to the training data
- Testing: Apply a statistical inference test to determine if an unseen instance belongs to the model
Assumption

Data is generated by a Gaussian process. Therefore it is normally distributed about the mean.

- Assume a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu$ and variance $\sigma^2$
- Estimate parameters $\Theta = \{\mu, \sigma\}$
- Labelling: Choose a threshold, e.g. $3\sigma$. Everything outside the normal range ($\mu \pm 3\sigma$) is an outlier.
- Scoring: Distance of data to the mean $\mu$
Univariate Models

Box Plot Rule

- **t-distribution**: distribution of the location of the true mean relative to the sample mean
- **t-test**: statistical hypothesis test; estimates confidence that test sample comes from the same distribution as training samples
- If $H_0$ is rejected, test sample is an anomaly
- Suitable for small sample sizes where normal behaviour can be easily quantified

**Student’s t-test**

- Probability Density Function (PDF)
  \[ P = \phi(\mu, \Sigma) \]
- **Mean**
  \[ \mu = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \]
- **Covariance**
  \[ \Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \]
**Mahalanobis Distance:** measures the distance between a point \( p \) and a probability distribution \( P = \eta(\mu, \Sigma) \):

\[
d(p, P) = \sqrt{(p - \mu)^T \Sigma^{-1} \cdot (p - \mu)}
\]

- Distance is zero if \( p \) is at the mean of \( P \)
- Distance grows as \( p \) moves away from the mean along each principal component axis
- Multi-dimensional generalisation of measuring how many standard deviations \( p \) is from the mean of \( P \).

---

You'll notice that the girls who are assigned to that particular duty are unusually tall. If the Germans were to somehow get their hands on the personnel records for all of the people who work at Bletchley Park, and graph their heights on a histogram, they would see a normal bell-shaped curve, representing most of the workers, with an abnormal bump on it—representing the unusual population of tall girls who we have brought in to work the plug boards.

— Neal Stephenson, *The Cryptonomicon*
A Mixture Model is a mixture of parametric statistical distributions. Most typically, the Gaussian Mixture Model (GMM) is a mixture of \( N \) independent Gaussian distributions. The contribution of each Gaussian to the mixture is determined by a weight vector \( \phi \):

\[
f(x, \Theta) = \sum_{i=1}^{N} \phi_i \cdot \eta(\mu_i, \Sigma_i)
\]

GMM is a mixture of \( N \) independent Gaussian distributions. The contribution of each Gaussian to the mixture is determined by a weight vector \( \phi \). The actual probability distribution of the entire data \( D \) is the prior probability that a data point is an anomaly. Initially all points are in \( M \).

\[
D = \lambda A + (1 - \lambda)M
\]

- Assume a GMM with two Gaussian mixtures:
  - \( M(\mu_1, \Sigma_1) \) for normal data
  - \( A(\mu_2, \Sigma_2) \) for anomalies
- \( D \) is the actual probability distribution of the entire data
- \( \lambda \) is the prior probability that a data point is an anomaly
- Initially all points are in \( M \)

Expectation (E) Step: Each point is assigned a probability of being in \( A \) based on how much the distributions change if the point is removed from \( M \) and added to \( A \).

Maximisation (M) Step: Points are assigned to \( A \) or \( M \) based on the (log-)likelihood function calculated in the E-step.

Iterate until we converge on the Maximum Likelihood Estimate (MLE), the parameters which provide the highest-probability explanation of the data.
- **Supervised Mode**: model normal and anomalous instances as separate mixtures
- **Semi-supervised Mode**: model only normal instances ("background") as mixtures
- Can be generalised to an arbitrary number of Gaussian mixtures
- Variational Bayesian algorithms can produce better results, e.g. Dirichlet Process Gaussian Mixture Model (DPGMM)

- Model structure is not defined *a priori*, it is determined from given data.
- More suitable where the underlying distribution of data is unknown, as it makes make fewer assumptions.
Yesterday I got a brilliant idea on how to implement automatic anomaly detection at CERN experiments. Today this work is done manually—many students/PhD students are looking at different distributions online. This is quite unreliable, and it's quite expensive, since you need many people to work all the time (nobody is paid for this—but you anyway spend money on travels). So, the basic idea is quite simple: one can bin each variable and look at distributions within each of bins. Knowing, that number of events observed inside each bin is Poisson-distributed, one can detect anomalies.

### Model
- Model is based on counting frequency of data assigned to pre-defined bins
- **Semi-supervised:** create a model of normal data only
- **Labelling:** If test data falls into a bin defined during training, it is normal, otherwise anomalous
- **Scoring:** Assign an anomaly score based on the height of the bin that test data is assigned to

### Advantages
- If the assumptions about the underlying model hold true, gives a statistically justifiable model for anomaly detection
- Anomaly score is associated with a confidence interval, which allows scoring of anomalies
- If the distribution estimation is robust to anomalies, statistical methods can be used in an unsupervised mode
Intr
oduction Global Anomalies Local Anomalies High-Dimensional Data Other Approaches

Overview

Statistical
Classification
Clustering

Classification-based Approaches

Disadvantages

- Assumptions may not hold true
- Not always obvious which test statistic is best
- Difficult to construct a hypothesis test for complex high-dimensional data
- Histogram-based techniques are easy to implement but cannot capture interactions between different attributes

Assumption

A classifier that can distinguish between normal and anomalous classes can be learned in the given feature space

- Training: Learn a model (classifier) from a set of labelled data instances
- Testing: Classify a test instance into one of the classes using the learned model

Classification-based Approaches

Supervised Multi-class Classification

- Training data contains instances belonging to one or more normal classes and the anomalous class

Semi-supervised Multi-class Classification

- Training data contains instances belonging to normal classes only
Intr
oduction Global Anomalies Local Anomalies High-Dimensional Data Other Approaches

Overview

Statistical Classification

Clustering

Summary

Classification-based Approaches

One-class Classification

- A single boundary is learned for all normal data

Rule-based Classifiers

- Learn rules from training data:
  \[ (A_1 = v_1 \land A_2 = v_2 \ldots) \implies C_1 \]
- Rule quality is evaluated by:
  - Support/Coverage
  - Confidence/Accuracy
- Labelling: test instances are matched to the best rule
- Scoring: take the inverse of the confidence score

Kernel Function-based Classifiers

Instance-based Learning
- Kernel function: a similarity function over pairs of data points
  \[ k : X \times X \to \mathbb{R} \]
- Typically \( O(N^2) \) complexity

Support Vector Machine (SVM)
- Kernel is a Radial Basis Function (RBF)
  \[ \phi(x, c) = \phi(||x - c||) \]
- Points outside the learned boundary are anomalies
Classification-based Approaches

Bayesian Approaches

- Suitable for Categorical Data
- One-class: Naïve Bayes
  
  $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

- Multi-class: Bayesian Networks

Advantages

- Testing phase is fast, as items are compared against a precomputed model

Disadvantages

- Model accuracy depends on the availability of accurately-labelled training data
  
  - Training data usually will not cover all possible types of anomaly
  
  - Training data often has to be labelled manually

- Classifiers usually only offer labelling, not scoring
Intr
oduction Global Anomalies Local Anomalies High-Dimensional Data Other Approaches,
Overview Statistical Classification Clustering Summary

Clustering-based Approaches

K-Means Clustering

- Partition data into \( k \) clusters
- Each point belongs to the cluster with the nearest mean

\[
\text{arg min}_s \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2
\]

- Parameter \( k \) is selected by the user
- Uses an iterative approach to converge to a local optimum

EM Clustering

- Normal data instances belong to a cluster in the data.
- Anomalies do not belong to any cluster.

- Apply a known clustering algorithm to the dataset
- Labelling: Instances which do not belong to any cluster are declared as anomalies
**Assumption**

Normal data instances lie close to their cluster centroid. Anomalies are far away from their cluster centroid.

- Apply a known clustering algorithm to the training data
- Compare the test data to the model of cluster centroids
- **Scoring:** the anomaly score for each data instance is the distance to its closest cluster centroid

**Advantages**

- Can be applied to any data type for which a clustering algorithm exists
- In supervised mode, test phase is very fast (number of clusters is small compared to number of data points)
- Can operate in unsupervised mode

**Disadvantages**

- Performance is highly dependent on how well the clustering algorithm captures the cluster structure of normal instances
- Techniques which detect anomalies as a byproduct of clustering are not optimised for anomaly detection
  - Some clustering algorithms force all points to be added to a cluster
  - Some techniques don’t work if anomalies are clustered
- Some clustering algorithms have high $O(N^2d)$ computational complexity
Introduction

Global Anomaly Detection
- Statistical Approaches
- Classification-based Approaches
- Clustering-based Approaches

Local Anomaly Detection
- Distance-based Approaches
- Density-based Approaches

Anomaly Detection in High-dimensional Data

Other Approaches

Local Anomaly Detection Approaches

- Nearest Neighbours
- Distance-based Approaches
- Density-based Approaches

Assumption

Normal data instances occur in dense neighbourhoods, while anomalies occur far from their closest neighbours

- Distance-based: Anomaly score is the distance of a data instance to its $k^{th}$-nearest neighbour
- Density-based: Anomaly score is the relative density of each data instance compared to its neighbourhood
Local Anomaly Detection Approaches

Nearest Neighbour-based Approaches

- All Nearest Neighbour-based approaches require a distance or similarity measure between pairs of data instances.
- Distance measure is usually required to be:
  - positive-definite (can’t have negative distances)
  - symmetric: \(d(a, b) = d(b, a)\)
- but is not usually required to satisfy the triangle inequality \(z \leq x + y\)
- A norm is a function that assigns a strictly positive length or size to each vector in a vector space.

- Euclidean distance

  In 2D Cartesian space:
  \[d(a, b) = d(b, a) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\]

  In 3D Euclidean space:
  \[d(a, b) = d(b, a) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}\]

  Generalisation to any number of dimensions is the Euclidean norm or \(L^2\) norm:
  \[d(a, b) = \|a - b\| = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}\]
Intr
oduction Global Anomalies Local Anomalies High-Dimensional Data Other Approaches

Overview

Near
est Neighbours

Distance-based Approaches

Density-based Approaches

Near
est Neighbours

Manhattan Distance

Minkowski Distance

- Minkowski Distance is a generalisation of Manhattan distance and Euclidean distance
- \( L^p \) norm:
  \[ d(a, b) = ||a - b||_p = \left( \sum_{i=1}^{n} |a_i - b_i|^p \right)^{\frac{1}{p}} \]
- \( p = 1 \) gives Manhattan distance
- \( p = 2 \) gives Euclidean distance

Distance-based Approaches

Distance to Nearest Neighbour

- Calculate distance of a data instance to its nearest neighbour
- Labelling: Threshold on the distance
- Scoring: Anomaly score = distance

\[ d(a, b) = ||a - b||_1 = \sum_{i=1}^{n} |a_i - b_i| \]

In the limiting case of \( p = \infty \), we get the Chebyshev distance

\[ L^\infty \) norm:

\[ d(a, b) = ||a - b||_\infty = \lim_{p \to \infty} \left( \sum_{i=1}^{n} |a_i - b_i|^p \right)^{\frac{1}{p}} = \max_{i=1}^{n} |a_i - b_i| \]

Chebyshev distance has been used as a distance measure for high-dimensional data. Fast to compute and accuracy comparable to \( L^1 \) or \( L^2 \) norms.

Distance measures with \( 0 < p < 1 \) have also been effective for high-dimensional data
Distance-based Approaches
Distance to Nearest Neighbour

- Calculate distance of a data instance to its nearest neighbour
- Labelling: Threshold on the distance
- Scoring: Anomaly score = distance
- Misses paired outliers

Distance-based Approaches
Distance to \( k \)-Nearest Neighbour

- Micro clusters \(|C| < k + 1\) become outliers
“Outliers are further away from the data”
Anomaly score = sum (average) of distances to k-Nearest Neighbours
More robust with respect to micro-clusters

Count the number of neighbours in a hypersphere of radius r.
- Fix radius \( r \) and use \( \frac{1}{n} \) as the anomaly score
- Fix \( n \) and use \( \frac{1}{r} \) as the anomaly score
For 2D data, the density of a data instance is:

\[
\frac{n}{\pi r^2}
\]

Anomaly score is the inverse of the density:

\[
\frac{\pi r^2}{n}
\]
**Introduction**

Global Anomalies

Local Anomalies

High-Dimensional Data

Other Approaches

**Overview**

**Distance-based Approaches**

**Density-based Approaches**

### Distance-based Approaches

- **Global Density**

  ![Diagram showing Global Density]  

  - However: k-NN Cannot handle variations in density

### Density-based Approaches

- **Relative Density**

  ![Diagram showing Relative Density]  

  - Global Density: Outlier \( o_1 \) will be detected, \( o_2 \) will not.
  - Relative Density: Anomalies are far from their neighbours, relative to the density of the local neighbourhood of each data point

- **Reachability Distance**

  ![Diagram showing Reachability Distance]  

  - Reachability Distance: Radius of the smallest hypersphere centred on the data instance that contains at least \( k \) neighbours (Here \( k = 4 \))
  - Local Density: 

  \[
  \pi \text{(hypersphere)}
  \]
Introduction Global Anomalies Local Anomalies High-Dimensional Data Other Approaches

Overview

Density-based Approaches

Local Outlier Factors (LOF)

- Choose parameter \( k \)
- Calculate the local density of all data instances
- The Local Outlier Factor (LOF) of a point \( p \) is the ratio:
  \[
  \text{LOF}_k(p) = \frac{\sum_{o \in N_k(p)} \text{ind}_k(o)}{|N_k(p)|}
  \]

\( \text{Normal instance: } \text{LOF}_k(p) \leq 1.0 \)

\( \text{Outlier: } \text{LOF}_k(p) \gg 1.0 \)

Michael Davis (CERN) Algorithms for Anomaly Detection 7 March 2017 78 / 119

Normal instance: local density of \( p \) is similar to that of its neighbours
Outlier: local density of \( p \) will be lower than that of its neighbours

Michael Davis (CERN) Algorithms for Anomaly Detection 7 March 2017 79 / 119

Density-based Approaches

LOF Variants

- Outlier Detection using In-degree Number (ODIN)
  - ODIN score is the number of \( k \)-NNs of \( p \) which have \( p \) in their \( k \)-NN
- Connectivity-based Outlier Factor (COF)
  - Add the next-closest instance to the NN (rather than the closest to \( p \))
  - Continue until we have \( k \) instances
- SLOM: LOF variant for detecting spatial anomalies in climate data
- LOF variant for categorical data using a similarity measure

Michael Davis (CERN) Algorithms for Anomaly Detection 7 March 2017 80 / 119

M. DAVIS

iCSC 2017
Introduction  Global Anomalies  Local Anomalies  High-Dimensional Data  Other Approaches

Distance- and Density-based Approaches

Advantages

- Unsupervised, data-driven approach
- Does not make any assumptions about the generative process that created the data, or the statistical distribution of the data
- Adapting to other types of data is straightforward: define an appropriate distance measure

Disadvantages

- Risk of misclassification
- Performance greatly relies on the distance measure chosen
  - Euclidian distances perform well...
  - ...but are expensive to compute
  - “Curse of Dimensionality”
- High \(O(N^2)\) computational complexity to calculate the neighbourhood

Computational Complexity

High \(O(N^2)\) computational complexity to calculate the neighbourhood can be mitigated by:

- **Indexing**: R-trees, R*-trees, X-trees can yield \(O(n \log n)\) complexity. But do not scale well in high dimensions.
- **Partitioning/Clustering**: Partition attribute space into a hypergrid. Linear in data size but exponential in number of attributes, so not suited to high-dimensional data.
- **Sampling/Pruning**: determine k-NN within small sample of dataset. Can result in incorrect anomaly scores if sample size is too small.

Anomaly Detection in High-dimensional Data

- “Curse of Dimensionality”
- Distance Concentration Effect
- Neighbourhood Selection
- Subspace Outlier Detection
- Outstanding Problems
**Curse of Dimensionality**

The term *dimensionality curse* is often used as a vague indication that high dimensionality causes problems in some situations. The term was first used by Bellman in 1961 for combinatorial estimation of multivariate functions. In the area of the nearest neighbors problem it is used for indicating that a query processing technique performs worse as the dimensionality increases.

—Beyer et al. 1999

**Distance Concentration Effect**

**Assumption**

The ratio of the variance of the length of any point vector (denoted by $\|X_d\|$) with the length of the mean point vector (denoted by $E[\|X_d\|]$) converges to zero with increasing data dimensionality.

**Consequence**

The proportional difference between the farthest-point distance $D_{\text{max}}$ and the closest-point distance $D_{\text{min}}$ (the relative contrast) vanishes.

—Beyer et al. 1999

If $\lim_{d \to \infty} \frac{\text{var}(\|X_d\|)}{E[\|X_d\|]} = 0$, then $\frac{D_{\text{max}} - D_{\text{min}}}{D_{\text{min}}} \to 0$.

- Relative contrast between near and far neighbours diminishes as the dimensionality increases.
- This is known as the concentration effect of the distance measure.
- It reduces the utility of the measure to discriminate between near and far neighbours.
Covers a broad range of data distributions and distance measures (generally: all integer $L^p$ norms with $p \geq 1$)
- Hinnenburg et al. show that $L^1$ and $L^2$ are the only integer norms useful for higher dimensions
- Aggarwal et al. show that fractional $L^p$ norms can be used, but the result is only valid for uniformly distributed data
- Effect can be partially countered by rescaling to unit dimensions ($\forall x, x_i \in [-1, 1]$)

However: for a constructed outlier ($x_i = 0.9$ in all dimensions), discrimination increases as dimensions increase:

This is because all dimensions add information
- Main problem for outlier detection in high-dimensional data is extra dimensions which do not add information
Neighbourhood Selection
What determines if the Nearest Neighbourhood is meaningful?

- High signal-to-noise ratio: irrelevant attributes mask the information in relevant attributes
- "Self-similarity Blessing": Latent correlation between the attributes results in an intrinsic dimensionality which is considerably lower than the representational dimensionality
- Concentration effect is less severe for clusters of points generated by the same mechanism

Pairwise Stability

- Pairwise stability between clusters holds when:
  - mean distance between points of different clusters ≥ mean distance between points of the same cluster
  - If clusters are pairwise stable, the NN of any point tends to belong to the same cluster
  - NN queries on the order of the cluster size can still be meaningful, even if differentiation between neighbours within the same cluster is meaningless
**Neighbourhood Selection**

**Pairwise Stability**

- Pairwise stability between clusters holds when:
  - mean distance between points of different clusters $\gg$ mean distance between points of the same cluster

- If clusters are pairwise stable, the NN of any point tends to belong to the same cluster

- NN queries on the order of the cluster size can still be meaningful, even if differentiation between neighbours within the same cluster is meaningless

**At high dimensions, a small change in radius $r$ leads to big change in volume of a hypersphere**

- Selecting neighbourhood using radius $r$ is unstable
- Hard to select correct value for $r$
- Selecting k-NN is more stable as it relies on a distance ranking rather than absolute distances

**Approximate Neighbourhoods**

- Anomaly detection in a reduced feature space:
  - Step 1: Global dimensionality reduction, e.g. by Principal Component Analysis
  - Step 2: Outlier detection in the reduced feature space
Anomaly detection in a reduced feature space:

- **Step 1:** Global dimensionality reduction, e.g. by Principal Component Analysis
- **Step 2:** Outlier detection in the reduced feature space
- Can be effective in selecting the neighbourhood
- However, the subspace is usually insufficient to derive all outliers
- Outlier detection step is likely to fail

**Johnson-Lindenstrauss Lemma**
- Proves that when \( n \) objects are projected into a lower-dimensional space of dimensionality \( O\left(\frac{\log n}{\epsilon^2}\right) \), the distances are preserved within a factor of \( 1 + \epsilon \)
- Reduced dimensionality does not depend on the original dimensionality
- Choice of error bound \( \epsilon \) gives a controlled trade-off between efficiency and precision
- Random projection is independent of the data and cheap to compute compared to PCA
Angle-based Outlier Detection (ABOD)

- Distance measure based on Cosine Distance
- **Normal data:** most other data objects are distributed in all directions
- **Outlier:** most other data objects are distributed in a few directions
- Lower variance signifies higher outlierness

Subspace-based Outlier Detection

- Grid-based clustering approach
- Clusters (as opposed to outliers) are not rare objects and are recognisable at higher dimensionality
- Outlierness based on how often the object is recognised as part of a cluster
- Relies on clusters being well-separated
- By finding clusters first, avoids data-snooping bias...
- …but grid-based approach leads to combinatorial explosion of the search space

Subspace-based Outlier Detection

- Data-snooping Bias
  - Say we choose $3\sigma$ as the threshold, then likelihood of an outlier is $0.9973^2$.
    - At $d = 10$, 97.33% of objects are within $3\sigma$
    - At $d = 100$, 76.31% of objects are within $3\sigma$
    - At $d = 1000$, 6.696% of objects are within $3\sigma$
  - At high dimensions, virtually every object is extreme in at least one dimension

Problems of High-dimensional Data

- **Concentration of Scores:** distances of attribute-wise i.i.d. objects converge to a normal distribution with low variance
- **Noise Attributes:** irrelevant attributes can mask relevant attributes
- **Definition of Reference Sets:** Need to know the neighbours to choose the subspace; need to know the subspace to find the neighbours
Intrduction Global Anomalies Local Anomalies High-Dimensional Data Other Approaches

Problems of High-dimensional Data

- **Data-snooping Bias**: Given enough subspaces, we can find at least one subspace in which the point is an outlier.
- **Exponential Search Space**: Number of possible subspaces grows exponentially with number of dimensions.
- **Thresholding**: While ranking outlier scores may be valid, it may be impossible to find a threshold between inliers and outliers due to low contrast.

Anomaly Detection in Structured Data

- **Regression Models**
  - Where we expect linearity in the data
- **Time Series Data**
  - Temporal sequence of data points, sometimes with cyclic patterns
- **Geographic/Directional Data**
  - Spatial relationship between data points
  - Spatial Outlier Factor, a variant of LOF based on spatial neighborhoods
- **Graph-based Data**
  - Social networks, transport networks, computer networks, ecosystems, …
  - Analysis of the adjacency matrix
  - Frequent subgraph mining
Intr
oduction Global Anomalies Local Anomalies High-Dimensional Data Other Approaches

Structured Data

Ensemble methods

- Geographic/Directional Data
  - Spatial relationship between data points
  - Spatial Outlier Factor, a variant of LOF based on spatial neighborhoods
- Graph-based Data
  - Social networks, transport networks, computer networks, ecosystems, …
  - Analysis of the adjacency matrix
  - Frequent subgraph mining

- Sometimes results can be improved by using more than one method, and combining the results
- Methods must exhibit two properties:
  - Accuracy
  - Diversity
- How to combine scores in a principled way?
  - Normalisation
  - Greedy Ensemble algorithms

Summary

- There are many different approaches to anomaly detection
  - Statistical
  - Classification- and Clustering-based
  - Distance- and Density-based
  - Structured Data
- Which algorithm is best will depend on:
  - Nature of the problem we wish to solve
  - Availability of labelled data
  - Data type(s)
  - Data distribution
  - Data dimensionality

Tools for anomaly detection algorithms

ELKI Data Mining Toolkit

- Algorithms for clustering and outlier detection
- Emphasis on unsupervised methods
- Includes data index structures for performance and scalability (e.g. R*-tree)
- Designed to allow easy and fair evaluation and benchmarking of algorithms
- Extensible: written in Java, released under AGPLv3 license
## References I

### General Books and Papers on Anomaly Detection


## References II

### References for Specific Slides

<table>
<thead>
<tr>
<th>Page</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Reproduced from <a href="https://xkcd.com/1781/">https://xkcd.com/1781/</a>, under the Creative Commons Attribution-NonCommercial 2.5 License</td>
</tr>
<tr>
<td>29</td>
<td>Reproduced from <a href="https://en.wikipedia.org/wiki/File:Warning_for_Gaussian_Distributions_(3539684489).jpg">https://en.wikipedia.org/wiki/File:Warning_for_Gaussian_Distributions_(3539684489).jpg</a>, under the Creative Commons Attribution 2.0 Generic License</td>
</tr>
<tr>
<td>36</td>
<td>Adapted from <a href="https://en.wikipedia.org/wiki/File:GaussianScatterPCA.svg">https://en.wikipedia.org/wiki/File:GaussianScatterPCA.svg</a>, under the Creative Commons Attribution 4.0 International License</td>
</tr>
<tr>
<td>41</td>
<td>Adapted from <a href="https://en.wikipedia.org/wiki/File:Csv-2d-gaussian-multivariate-distributions.svg">https://en.wikipedia.org/wiki/File:Csv-2d-gaussian-multivariate-distributions.svg</a> (public domain)</td>
</tr>
<tr>
<td>49</td>
<td>Alex Rogozhnikov, <em>Anomaly detection at CERN experiments</em>, blog post at Brilliantly Wrong, 10 May 2015</td>
</tr>
<tr>
<td>56</td>
<td>Reproduced from Chandola et al. 2009</td>
</tr>
<tr>
<td>57</td>
<td>Reproduced from Chandola et al. 2009</td>
</tr>
<tr>
<td>59</td>
<td>Reproduced from <a href="https://en.wikipedia.org/wiki/File:Kernel_Machine.svg">https://en.wikipedia.org/wiki/File:Kernel_Machine.svg</a>, under the Creative Commons Attribution-Share Alike 4.0 International License</td>
</tr>
<tr>
<td>60</td>
<td>Reproduced from <a href="https://en.wikipedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg">https://en.wikipedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg</a>, under the Creative Commons Attribution 3.0 Unported License</td>
</tr>
</tbody>
</table>

---

Michael Davis (CERN) Algorithms for Anomaly Detection 7 March 2017 112 / 119
Intr
oduction

Global Anomalies

Local Anomalies

High-Dimensional Data

Other Approaches

References IV


84–93 Adapted from Erich Schubert and Eirini Ntoutsi, Data Mining Tutorial, Session 6: Outlier Detection, Ludwig–Maximilians–Universität München, 2012


100 V.Hautamaki, I.Karkkainen and P.Franti, Outlier detection using k-nearest neighbour graph, Proceedings of the 17th International Conference on Pattern Recognition, IEEE 2004

References V

J.Tang, Z.Chen, A.W.Chee Fu and D.W.Cheung, Enhancing Effectiveness of Outlier Detections for Low Density Patterns, Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2002

P.Sun and S.Chawla, SLOM: A new measure for local spatial outliers, Knowledge Information Systems 9, 4, 2006


A.Hinneburg, C.C.Aggarwal and D.A.Keim, What is the nearest neighbor in high dimensional spaces?, Proceedings of the 26th International Conference on Very Large Data Bases—VLDB 2000

C.C.Aggarwal, A.Hinneburg and D.Keim, On the surprising behavior of distance metrics in high dimensional space, Proceedings of the 8th International Conference on Database Theory—ICDT 2001

References VI


121 Reproduced from https://en.wikipedia.org/wiki/File:GaussianScatterPCA.svg, under the Creative Commons Attribution 4.0 International License

122 W.B. Johnson and J.Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, Conference in Modern Analysis and Probability, AMS 1984

123 Timothy de Vries and Sanjay Chawla, Finding Local Anomalies in Very High Dimensional Space, Proceedings of the IEEE International Conference on Data Mining—ICDM 2010

125 Hans-Peter Kriegel, Matthias Schubert and Arthur Zimek, Angle-based Outlier Detection in High-dimensional Data, Proceedings of the 14th SIAMKDD International Conference on Knowledge Discovery and Data Mining—KDD ’08

References VII

127 Emmanuel Müller, Ira Assent, Uwe Steinhoosen and Thomas Seidl, OutRank: ranking outliers in high dimensional data, Proceedings of the 24th International Conference on Data Engineering (ICDE) Workshop on Ranking in Databases (DBRank), 2008
Applying natural evolution for solving computational problems

First lecture: Introduction to Evolutionary Computation
Second lecture: Genetic Programming

Inverted CERN School of Computing 2017
Daniel Lanza - CERN

Agenda
• Introduction to Evolutionary Computation
  • Introduction to Evolutionary Algorithms
  • Use cases, practical examples
  • Key concepts
  • Representation of individuals
  • Phases
  • Evolutionary Computation research tool (ECJ)

Introduction to Evolutionary Algorithms

An evolutionary algorithm (EA) is a heuristic optimization algorithm using techniques inspired by mechanisms from organic evolution such as mutation, recombination, and natural selection to find an optimal configuration for a specific system within specific constraints. [1]

• History
  • Idea originated in the 1950s
  • L. Fogel 1962 (San Diego, CA): Evolutionary Programming
  • I. Rechenberg & H.-P. Schwefel 1965 (Berlin, Germany): Evolution Strategies

• When to use them
  • When finding exact solution is computationally too demanding, but near-optimal solution is sufficient

Use cases, practical examples

• Aircraft wing design [8][9]

D. LANZA GARCIA

A. 62

iCSC 2017
Use cases, practical examples

• Electronic circuit design, known as evolvable hardware [2]

• Wireless sensor/ad-hoc networks [4].

• Vehicle routing problems (traveling salesman problem) [3].

• Feature selection used by machine learning algorithms. [6]

• Image processing (face recognition) [5].

• ... [7]

Key concepts

• A population of individuals is evolved through generations

• Each individual's genome describes a candidate solution

• The fitness function evaluates new individuals

• The evolutionary process is finished as soon as the optimal solution is found
Representation of individuals

• It should be able to represent all the search space
• But should not be able to represent impossible solutions

• Examples of solutions to different problems:
  
  \[1, 0, 1, 0, 1, 1, 1, 1\]
  
  \[4.5, 1, 100.3, 9, 21, 934, 1\]
  
  \["right", "left", "up", "up", "left", "left", "down"\]
  
  \[4.5, "left", false, true, 9\]

Phases

• Similarly to how natural evolution works...

  - **Initialization**: Randomly generated
  - **Evaluation**: Calculate fitness for each individual
  - **Breeding**: Individuals are crossed over and mutations take place
  - **Selection**: Choose individuals for breeding
  - **Optimal solution**

Phases: in detail by using an example

• “MaxOne” problem
  
  • Starting from randomly generated strings of 0s and 1s
  • Evolve to the optimal solution, a string of 1s

  \[1, 1, 1, 1, 1, 1\]
Phases: initialization

- The first population is filled up with individuals.
- The individuals are randomly generated following problem's criteria.
- The population size is determined beforehand and remains fixed.

Phases: evaluation

- Practically the fitness function defines the problem.
- Get the fitness of each individual.
- Fitness describes how well the individual solves the problem.
- In "MaxOne", fitness is defined as the number of 1s in the individual.

Phases: selection

- A selection strategy is defined to choose the parents.
- Selected individuals, parents, will be used for breeding.
- Fitness is taken into account (best individuals).
- But also some randomness affects the selection (simulating real life).
- Individuals with reduced fitness could have valuable features.
- Elitism (optional): best individual is copied to the next generation.

Phases: selection (techniques)

- Tournament [11]
  - A fixed number of individuals are randomly selected.
  - Among them, the best one is selected.
**Phases: selection (techniques)**

- Roulette-wheel [12]
  - Selection probability is proportional to individual's fitness

- Others... [13]

**Phases: breeding**

- Selected individuals cross-over
- New individuals fill up the next generation of individuals
- Selection and breeding phases are performed till next population is filled

**Phases: breeding (techniques)**

- Some times features cannot be mixed
  - Smarter operations need to be applied
  - For example:
    - Parents: [4.5, "left", false, true, 9] [1.2, "right", true, true, 9]

- Depends on the problem
  - Number: average, max, min, sum, ...
  - Boolean: and, or, xor, ...
  - Strings: concatenate, replace, remove, split, ...

**Phases: mutation**

- With a very little likelihood
- A random modification is applied
Phases: mutation (techniques)

- Any modification can be considered a mutation

\[
\begin{bmatrix}
1, 0, 0, 1, 0, 1, 1
\end{bmatrix} \times \begin{bmatrix}
1, 0, 0, 1, 0, 1, 1
\end{bmatrix} = \begin{bmatrix}
1, 1, 0, 1, 0, 0, 1
\end{bmatrix}
\]

- More complex genomes could have their features modified to any possible value

\[
\begin{bmatrix}
4.5, \text{"left"}, \text{false}, \text{true}, 9
\end{bmatrix} \times \begin{bmatrix}
0.5, \text{"right"}, \text{false}, \text{true}, 9
\end{bmatrix} = \begin{bmatrix}
4.5, \text{"right"}, \text{false}, \text{false}, 9
\end{bmatrix}
\]

Phases: evaluation, selection, breeding, ...

- The loop keeps going till the optimal individual is found

Evolutionary Computation research tool (ECJ)

- Developed at the George Manson University [10]
- Eliminates the need of implementing the evolutionary process
- Highly used in the community
- Main features:
  - Multi-platform: Java
  - Flexibility: easy to implement many kind of problems
  - Configuration files
  - Check points
  - Multi-threading
  - Pseudo-random number generator: reproduce results

Evolutionary Computation research tool (ECJ)

- Code architecture allows pluggable and customized components
- Several built-in implementations for every component
Evolutionary Computation research tool (ECJ)

• Configuring the MaxOne problem

```java
breedthreads = 1
evalthreads = 1
seed.0 = 4357
state = ec.simple.SimpleEvolutionState
pop = ec.Population
init = ec.simple.SimpleInitializer
finish = ec.simple.SimpleFinisher
eval = ec.simple.SimpleEvaluator
stat = ec.simple.SimpleStatistics
exch = ec.simple.SimpleExchanger
generations = 200

pop.subpops = 1
pop.subpop.0 = ec.Subpopulation
pop.subpop.0.size = 10
pop.subpop.0.species = ec.vector.BitVectorSpecies
pop.subpop.0.species.fitness = ec.simple.SimpleFitness
pop.subpop.0.species.ind = ec.vector.BitVectorIndividual
pop.subpop.0.species.genome-size = 20
pop.subpop.0.species.mutation-type = flip
pop.subpop.0.species.mutation-prob = 0.01
pop.subpop.0.species.pipe = ec.vector.breed.VectorMutationPipeline
pop.subpop.0.species.pipe.source.0 = ec.vector.breed.VectorCrossoverPipeline
eval.problem = ec.app.tutorial1.MaxOnes
```

• Implementing the MaxOne fitness function: `ec.app.tutorial1.MaxOnes`

```java
public class MaxOnes extends Problem implements SimpleProblemForm {
    public void evaluate(Individual ind,
            final int subpopulation,
            final int threadIndex) {
        int sum = 0;
        BitVectorIndividual vecInd = (BitVectorIndividual) ind;
        for (int i = 0; i < vecInd.genome.length; i++)
            sum += (vecInd.genome[i] ? 1 : 0);
        (SimpleFitness vecInd.fitness).setFitness(Steady.
                sum / (double) vecInd.genome.length,
                sum == vecInd.genome.length);
        vecInd.evolved = true;
    }
}
```

Evolutionary Computation research tool (ECJ)

• Execution

```java
// Sample code for running ECJ
public static void main(String[] args) {
    // Configuration code...
    // Evaluation code...
    // Display results...
}
```

Questions?
Applying natural evolution for solving computational problems

First lecture: Introduction to Evolutionary Computation
Second lecture: Genetic Programming

References

4. BINET76 – Distributed Software Systems Group, University of Massachusetts, Boston
5. D. Lanza, F. Chavez, F. Fernandez, C. Benavides-Alvarez and J. Villegas, Speeding up Evolutionary Approaches to Face Recognition by Means of Hadoop. EVO 2016
Applying natural evolution for solving computational problems

First lecture: Introduction to Evolutionary Computation
Second lecture: Genetic Programming

Agenda
• Genetic Programming
  • Introduction to GP
  • Representation of individuals
  • Phases
  • The problem of bloat
  • Implementing GP with ECJ
  • Distributed processing

Introduction to Genetic Programming
Genetic programming (GP) is a technique whereby computer programs are encoded as a set of genes that are then evolved using an evolutionary algorithm. The space of solutions consists of computer programs. [1]

• Belongs to the class of evolutionary algorithms

• History
  • J. Holland 1962 (Ann Arbor, MI): Genetic Algorithms
  • J. Koza 1989 (Palo Alto, CA): Genetic Programming

• When to use them
  • When finding exact solution is computationally too demanding, but near-optimal solution is sufficient

Representation of individuals
• Main characteristic of GP
  • Individuals represent computer programs

• Individuals are represented as trees
  • Nodes: operations
  • Terminals: values or variables

\{(a \times b) + \{c / 6\}\}
Phases

• Following the evolutionary process

Initialization
Randomly generated

Evaluation
Calculate fitness for each individual

Breeding
Individuals are crossed-over and mutations take place

Selection
Choose individuals for breeding

Optimal solution

Phases: initialization

• First population is filled up with individuals
• Randomly generated trees with allowed operations and terminals
• Initial individual size is limited to a range of values

Phases: evaluation

• Computer programs represented by individuals are executed
• Different or all possible inputs are tried and output is checked
• Fitness could be the percentage of correct outputs

Phases: selection

• A selection strategy is defined to choose the parents
  • Parents are individuals that will be used for breeding
  • Fitness is taken into account (best individuals)
  • But also some randomness affects the selection (simulating real life)
    • Individuals with reduced fitness could have valuable features
  • Elitism (optional): best individual is copied to the next generation
  • Other factors that could be taken into account (multi-objective selection):
    • Tree size
    • Computational cost
**Phases: breeding**

- Selected individuals are crossed-over
- New individuals fill up the next generation of individuals
- Selection and breeding phases are performed till next population is filled

**Phases: mutation**

- With a very little likelihood
- A random modification is applied

**Phases: evaluation, selection, breeding, ...**

- The loop keeps going till an individual provides a proper solution

**The problem of bloat**

- We would like solutions:
  - Understandable by humans, therefore simple
  - Computationally cheap to execute (CPU and memory)

- Bloat: the continuous increment in size of the trees
- Bigger trees use to provide “better” solutions to the problems
- Control mechanisms need to be applied
The problem of bloat: control mechanisms

- Limited tree size [3]
- Size punishes fitness [4]
- Multi-objective selection techniques [5]
- Eliminate introns (code that does nothing) [6]

- Computational time to evaluate can be consider as size
  - That would include:
    - The complexity of used operations
    - The amount of operations
  - Normally time is hard to obtain
    - Other processes may interfere
  - Implicit methods [7] could be applied, no need to measure time

Evolutionary Computation research tool (ECJ)

- Developed at the George Manson University [2]
- Eliminate the need of implementing the evolutionary process
- Highly used in the community
- Main features:
  - Multi-platform: Java
  - Flexibility: easy to implement many kind of problems
  - Configuration files
  - Check points
  - Multi-threading
  - Pseudo-random number generator: reproduce results

Evolutionary Computation research tool (ECJ)

- Configuring the multiplexer problem

```
parent0 = ~/.gp/kosko/kosko.params
# Function set
gp.fs.size = 1
gp.fs.0.name = f0
gp.fs.0.size = 10
# Define problem
eval.problem = ec.app.multiplexer.Multiplexer
eval.problem.data = ec.app.multiplexer.MultiplexerData
eval.problem.bits = 2
```
Evolutionary Computation research tool (ECJ)

- Problem implementation

```java
public class MultiplierData extends GPData {
  // Problem implementation
}
```

Evolutionary Computation research tool (ECJ)

- Functions implementation

```java
public class MultiplierData extends GPData {
  // Functions implementation
}
```

Evolutionary Computation research tool (ECJ)

- Terminals implementation

```java
public class AI extends GPNode {
  // Terminals implementation
}
```

Evolutionary Computation research tool (ECJ)

- Execution

```java
public class AI extends GPNode {
  // Execution
}
```
Distributed processing

- Evaluation of individuals could be computationally expensive
- Easily parallelizable
- Different approaches to distribute the work
  - Island models
  - Master-slaves
  - Integrating ECJ with Hadoop [8]

Distributed processing: an example [9]
Summarizing

• EA and GP can be applied for a wide range of applications
  • When finding the exact solution is computationally expensive
  • When the near-optimal solution is sufficient

• Similar to the evolution of individuals in nature, computational problems can be solved through evolution
  • Initialization, evaluation, selection, breeding and mutation

• Different challenges still to face
  • Bloat control
  • Expensive evaluation, distribution of workload

• Existing tools, like ECJ, help with the research and application

Questions?

References

Distributed consensus and fault tolerance

Lecture 1 / 2

Georgios Bitzes, CERN
iCSC 2017

Let's talk about distributed systems

Lecture 1
- Introduction to distributed systems
- Replication and split-brain
- Strong vs eventual consistency
- The raft consensus algorithm

Lecture 2
- Two Generals, Byzantine Generals
- Byzantine fault tolerance
- Bitcoin and blockchain consensus

Distributed systems are all around us

- Infrastructure:
  - Networking, routing algorithms
  - Flight control systems
  - Banking systems, ATMs

- Internet services: running these on a single machine would be unthinkable
  - Facebook, Twitter
  - Google search, gmail
  - Github

- But the need for distributed systems appears long before we have to scale to millions of users
Scalability

- As the load increases, one server no longer enough to serve all clients
- We’re bound to hit some bottleneck
  - CPU power
  - Memory size
  - Memory bandwidth
  - IO operations per second
  - Network bandwidth
  - Software, more often than not cannot fully exploit our powerful hardware

Latency

With only a single server, some clients will suffer from high latency

Fault tolerance

- Our system has a single point of failure – one faulty hard drive and the service goes down
- Downtime of certain critical services can cause great disruption: credit card processing, air traffic control, or… stackoverflow.com

Distributed systems to the rescue

- A distributed system solves the above problems nicely
  - Scalable performance: add more machines as needed
  - Lower latencies: clients connect to the server nearest to them
  - Fault tolerance: if a machine goes down, the others can detect it and take over its responsibilities
- Caveat: distributed systems add a lot of complexity
What is a distributed system?

- Distributed system: A collection of independent computers that appears to its users as a single, coherent system
- Nodes coordinate by exchanging messages through the network

Distributed system: added complexity

Making a system distributed adds some complexity:
- **Unreliable network**: messages get lost, delayed, re-ordered, corrupted
- **Node failures**: the more machines we have, the higher the probability of some failing
- **Latency**: communication far slower than using local shared memory or IPC
- **Limited bandwidth**
- **Security**: “is this message really coming from who I think it is”?

Multiple single points of failure?

- What if every node depended on every other for its correct operation?
- A single failure will bring them all down
- Just because a system is distributed, doesn’t mean it’s fault tolerant

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.”

Leslie Lamport
Replication as a means to fault tolerance

- A database machine holding 25% of all our data crashes and burns – hard drive is \textit{unrecoverable}
- What happens next? Some possibilities...
  - No backups, \textit{data lost forever} – oops. Not acceptable, will create bad publicity, erode user trust
  - A backup taken the previous day is manually restored. Much better, but new user data in the last 24h is lost, and \textit{long downtime} during manual restore
  - A replicated database machine takes over immediately after the failure – no data loss, no downtime, \textit{users don't even notice}

Key-value stores (used in future examples)

- Simple database, clients can perform 2 operations
  - \textit{write} a value into a key
  - \textit{read} it back.

```
127.0.0.1:6379> SET favorite_food pickles
OK
127.0.0.1:6379> GET favorite_food
"pickles"
127.0.0.1:6379> SET favorite_language c++
OK
127.0.0.1:6379> GET favorite_language
"c++"
```

A naive replication protocol

- Assume we want our key-value store replicated on 3 nodes.
- Let’s invent our own simple replication protocol:
  1. A client sends a write request: propagate the change to all other sibling nodes. Ignore errors.
  2. A client does a read: give back the local value stored on the contacted node
  3. On receipt of a propagated change from a sibling node: simply apply it by updating the local value.
A naive replication protocol (2)

- In simple cases and good network conditions, this protocol might appear to work.
  - https://gbitzes.github.io/icsc(animation/#naiverep)
- What if a node goes down for maintenance for 5 minutes?
  - All writes within that window are not replicated onto it
- What if certain nodes receive the updates in a different order?

The problem: Split brain

- Our naive protocol will inevitably lead to split brain
- What we want: replicated nodes to agree on the state of the key-value store, that they all come to a consensus about each update
- We’ll talk about a correct algorithm later

Failure model

The failure model we’ll concern ourselves for now:

1. Fail-recover faults
   - A node goes down, stops responding to messages.
   - The other nodes can detect this through timeouts – “if 127.0.0.10 doesn’t respond in 100ms, it’s down”
   - … but failed nodes can recover
2. Delayed / lost messages: Messages between nodes can be arbitrarily delayed or lost, but not corrupted

- Later on: byzantine faults
Network partitions

- Tolerance to delayed and lost messages implies tolerance to network partitions
- **Network partition**: parts of the system become disconnected from each other

Network partitions (2)

- Problematic because each part is making decisions *independently*
- What if clients try to write different values to each partitioned part? Which value is “correct”?

**Strong vs eventual consistency**

CAP theorem

- When faced with *Network Partitions* (P), a distributed system can be either **Consistent** (C) or **Available** (A)
Strong consistency

- System appears **externally consistent** to clients
- Requests are **refused** if consistency cannot be guaranteed

Eventual consistency

- Best-effort: try to be consistent, but without guarantees
- Internal inconsistencies **may be exposed** to clients

Strong consistency (2)

- Internal, temporary inconsistencies: **inevitable** in distributed systems
- Strong consistency: internal inconsistencies are resolved and **not exposed** to clients
- The cost: **sacrifice** availability in favor of consistency

Eventual consistency (2)

- Conflicts are resolved after partition heals
- Common approach: **"last writer wins"**
- Received "pizza" at t = 16783 ms
- Received "pickles" at t = 16787 ms
- Yours is newer, agreeing on "pickles"
Eventual consistency (3)

- Clients may receive stale values

Which one is better?

- Depends on the application, both are useful
  - Eventual consistency: generally more performant and scalable
    - example: DNS, Amazon S3
  - Strong consistency: safety guarantee, every read receives the most recent write or error
    - example: certain SQL databases

Question

Which form of consistency does our naive replication protocol provide?

1. Strong consistency
2. Eventual consistency
3. Neither

- Correct answer: neither.
- In our protocol, inconsistencies are never resolved – not even "eventually"
Distributed consensus

- A number of nodes come to an **agreement** about a **value**
- Any node can **crash** or **recover** at any time

Raft consensus algorithm

**Master – slave replication**

- One of the nodes is elected to become the **master** (or **leader**)
Leader failover

- The leader sends regular heartbeats to all followers
- If a follower stops receiving heartbeats, it assumes leader failure and triggers an election
- An election is won if a node receives positive votes from at least a majority of the cluster

Heartbeats

- Haven’t heard from the leader for 2 sec… Something is wrong

Election terms

- A leader is elected for a specific election term based on majority vote
- Only one leader can be elected per term
- Possible to have multiple leaders at a time, due to network partitions – all will have different terms
- The leader with the highest term “wins” and can override decisions made by the others

Leader election

- A successful election: 2 out of 3 nodes agree on the new leader
- Vote for me for term 20
- Vote for me for term 20
- Vote granted
- Used to be leader for term 19
Bringing outdated nodes up-to-date

- One of the followers goes offline for 10 minutes – how to bring it up-to-date?
- Record all writes into an indexed log, and replicate it
- Contains also the term during which the entry was recorded

<table>
<thead>
<tr>
<th>Index</th>
<th>Term</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>SET food pizza</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>SET language c++</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>SET food pickles</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>SET answer_to_life 42</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Log replication

Hey, are you there?

Yeah, my last index is #149

You're missing entries 150 to 203..

Here they are…

Application of log entries

- Log entries represent changes to the database (also called the state machine)
- At some point, these changes have to be applied to the state machine

State machine replication

- Not safe to apply an entry immediately, since in rare cases it might get rolled back by a subsequent leader
- Log entries are considered committed once a majority of nodes have them
- Once an entry is committed, it’s guaranteed that it won’t be rolled back, and can be safely applied to the state machine
- Only nodes that have all committed entries can ever succeed during a leader election
Summary

- Distributed systems are **necessary** to provide scalability, low latency, and fault tolerance.
- Replicated nodes can offer transparent failover, but be careful of **split brain**.
- Several approaches to replication, need to compromise between **consistency**, **availability**, **scalability**, and more.
- The *Raft consensus algorithm* offers a good basis for a strongly consistent distributed system.

Bedtime reading

- PACELC theorem, extension to CAP theorem
- The raft paper
  [https://raft.github.io/raft.pdf](https://raft.github.io/raft.pdf)
- Paxos algorithm
- Strong consistency models
  [https://aphyr.com/posts/313-strong-consistency-models](https://aphyr.com/posts/313-strong-consistency-models)

Thanks

Any questions?

See you in Lecture 2
Georgios.bitzes@cern.ch
Let's talk about distributed systems

Lecture 1
- Introduction to distributed systems
- Replication and split-brain
- Strong vs eventual consistency
- The raft consensus algorithm

Lecture 2
- Two Generals, Byzantine Generals
- Byzantine fault tolerance
- Bitcoin and blockchain consensus

The two generals paradox
- Two armies have surrounded a city
- Their generals must decide together whether to attack or retreat
- Communication through messengers, must pass through the city and might be intercepted
- Both must take the same decision

Two Generals, Byzantine Generals

Easy enough... or not?

We attack tomorrow at dawn.

Did they get my message? I can't attack just yet.

Agreed, we attack tomorrow at dawn.

Did they get the confirmation? If not, they won't attack...

Confirmation has been received.

Did they get my confirmation? If not, they won't attack...
The two generals paradox (2)

- No protocol exists that guarantees both generals are 100% certain of the decision of the other
  - Proofs exist
- After 500 confirmations, both would be pretty sure the other will attack
- But “pretty sure” is not guaranteed 100% certainty

The two generals paradox (3)

Simplified impossibility proof:

- Let’s assume a protocol that exchanges N messages exists, which guarantees certainty
- The Nth message could be lost… meaning, the first N-1 messages must be sufficient to guarantee certainty
- Therefore, there exists such a protocol which exchanges N-1 messages
- Absurd conclusion: there exists such a protocol that exchanges 0 messages

Absurd conclusion: there exists such a protocol that exchanges 0 messages
The problem of Byzantine Generals

- Generalization of Two Generals
- N generals, decision to attack or retreat based on *majority vote*
- Some generals might be secretly traitors, and try to *manipulate* the vote…
- **Goal**: achieve consensus between honest nodes

The problem of Byzantine Generals (2)

- #1 wants to *attack*
- #3 wants to *retreat*
- #1 receives *retreat* votes from #2 and #3
- #3 receives *attack* votes from #1 and #2
- Result: #3 attacks *alone*, #1 retreats

Raft overview

In the previous lecture, we saw the main points of raft
1. Replicated log of all write operations

<table>
<thead>
<tr>
<th>Index</th>
<th>Term</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>SET food pizza</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>SET language c++</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>SET food pickles</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>SET answer_to_life 42</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

2. Leader node elected by majority ensures correctness
3. Log entries committed once replicated on a majority
Back to failure models

Reminder – our failure model so far has been:

1. **Fail-recover faults**
   - A node goes down, stops responding to messages.
   - The other nodes can detect this through timeouts – “if 127.0.0.10 doesn’t respond in 100ms, it’s down”
   - … but failed nodes can recover
2. **Delayed / lost messages**: Messages between nodes can be arbitrarily delayed or lost, but **not** corrupted

Even more failures

Clearly, our imagination so far was lacking. What if…

- the network starts corrupting messages?
- an attacker intercepts and modifies inter-node communication?
- some of the nodes are …evil?

Byzantine fault tolerance

- **Byzantine fault**: Any kind of conceivable fault, like…
  - corrupted messages
  - nodes behaving differently depending on **who** they’re talking to
  - nodes **conspiring** with each other to cause disruption

How can you make a reliable computer service? […] It may be difficult if you can’t trust anything and the entire concept of happiness is a lie designed by unseen overlords of endless deceptive power.

The Saddest Moment, James Mickens
Let’s break raft…

- Raft is not byzantine fault tolerant – countless opportunities for an evil node to cause mayhem
- A byzantine leader can change clients’ requests

Let’s break raft… (2)

- A byzantine node can trivially claim it’s the leader, and all other nodes will believe that
- Doesn’t have to prove it won an election!

Let’s break raft… (3)

Or continuously trigger elections, never allowing any leader to emerge – cluster becomes *useless*

Various causes of byzantine faults

- **Benign causes**
  - Corruption of network packets, disk data
  - A random *memory bitflip* causes a node to act unpredictably
  - A rare bug in the code causes a node to violate protocol

- **Malicious causes**
  - An attacker changes *inter-node traffic*
  - An attacker *takes over* a node
Possible counter-measures

- **Cryptographic signing, checksums**: can help ensure authenticity and integrity of messages, data

- **ECC memory**: can fix or at least detect memory errors

- Very difficult to protect against all kinds of faults… some protocols exist
  - all have their limitations

Reality check

- Is such level of paranoia justified?

- When running 50,000 servers, what if one “goes byzantine”?

- Could it disrupt some of the rest? Maybe…

- Good idea to at least protect against some byzantine faults

What is bitcoin?

- A currency not influenced by any central authority
- A currency based on cryptography, not human trust

- A distributed network of nodes processing transactions, **highly resilient** to byzantine attacks
  - **Everybody** can run their own bitcoin node
  - … meaning **nobody** can be fully trusted
Short history of bitcoin

- **November 2008**: “Satoshi Nakamoto” publishes a paper on a “peer-to-peer electronic cash system”
- **January 2009**: The bitcoin network comes online, with “Satoshi” mining the first block
- **2011 – 2016**: Bitcoin keeps gaining popularity… alternative cryptocurrencies appearing
- **September 2016**: The number of bitcoin ATMs worldwide reaches 770

Bitcoin addresses

- Anybody can generate one from a simple public-private key pair
  - **Public**: 1DwAdnhHZ3w2Eowo6SPZZaGNMXVJwbdGZNY
  - **Secret**: L3H8Yz7YPshShw5yjTH72c1y12UnUerFnzd9wjEDyhs1X5bUW5q
- Anyone with the public address can **send** bitcoins to that address
- To **spend** them, need to have the secret

A bitcoin transaction (super simplified)

- Inputs: The outputs of *previous* transactions
  - Output #2 of transaction 3b96bb7e197e...
  - Output #1 of transaction e1afd89295b68...
  - Output #6 of transaction e79fe1dad370e...
- Outputs
  - 0.1 BTC to address 1HmxmBAX413yGY2LDoEN8FHBok61aT4w2d
  - 0.2 BTC to address 1Hozk3UFZd5Gd7PENAvg33ouFHzvsCFJ7
- Signatures proving **ownership** of input addresses

Is this enough? No!

- Seller must be able to verify the transaction is **valid**
- The **input sources** might not exist, or buyer might not have the **right to spend** them (invalid signature)
The bitcoin blockchain (super simplified)

- A replicated log of all bitcoin transactions since the beginning of time
- Each block contains a list of signed transactions, but also the SHA256 hash of its previous block

Transaction broadcasting

“I declare I want to send 0.1 BTC from 12vEsHqQ3XLuk2Mpt5hLEQg2pkeweQWw4f to 16yZEEnG3XK6hH30hH30hH30hH30hPUMpWu72c, and here’s a signature to prove I own these bitcoins. Please add my transaction to your blockchains”

Is this enough? Still no

- The new transaction must end up on the blockchain of all bitcoin nodes
- Everybody on the network must be aware of the transaction otherwise seller can’t use the coins

Is this enough? Maybe...

- It’s not enough that one node confirms the transaction
- The seller must be sure the entire network confirms it
  - if not, they won’t be able to spend the coins in the future
Reaching blockchain consensus

- Bitcoin nodes will receive the transactions with a **different order**
- **Critical** that nodes agree on the order
- **Double-spending**: Two transactions that spend the same input source – only the first can be valid
- Everyone must reach consensus on which is first, and which is invalid

Blockchain forks

- By definition, the “true” blockchain is the longest one
- An honest node notified of a longer blockchain must immediately switch to it

Transactions on “orphaned” blocks have essentially been **rolled back**

Blockchain forks (2)

- “Longest blockchain wins”: all bitcoin nodes converge rapidly to a common blockchain
- The more blocks there are ahead in the chain, the less likely our transaction gets rolled back
- Extra blocks serve as additional **confirmations** that bitcoin nodes acknowledge our transaction
  - they confirm it as valid
  - they confirm it to be part of the currently longest blockchain they know

Is this enough? Almost…

- The seller can cryptographically confirm there are 15 further valid blocks after the one with the transaction
  - reasonably certain that any forks will contain it… or not?

I want bread, here’s a transaction to your address with 0.1 BTC. I have broadcasted it already

OK, indeed the transaction reached my blockchain with 15 added confirmations. Here’s your bread
A byzantine attack

Something is missing… the following attack will succeed:

1. Publish a transaction, which is added to block #190
2. Wait for 20 confirmations
3. Seller is reasonably sure the network reached consensus on this transaction
4. Seller gives you bread
5. Publish a new blockchain, forking from node #189 with 30 empty blocks after it
6. Network switches to your blockchain, since it’s longer
7. You get to keep both the bread and the coins

51% attacks

- Bitcoin nodes are constantly trying to add new blocks – rewards for those that succeed
- An attacker must outrun them and produce blocks more quickly than the rest of the network combined
- Only possible for attackers who hold more than 50% of the total SHA256 computational power of the entire network
  - outrageously expensive to pull off

Proof of work

- Solution: make it expensive to add a new block
- The SHA256 of a block must start with a certain number of zeroes, otherwise honest nodes reject it
- The only way to produce such a block is brute force
  - The block format contains a counter for this purpose, which influences its resulting SHA256
  - Keep incrementing until you get lucky

The SHA256 computation arms race

- Best CPUs: 10 – 100 million hashes per sec
- Best GPUs: 100 – 1000 million hashes per sec

Specialized bitcoin mining hardware: FPGA, ASIC
- An ASIC released in 2012: 60,000 M hashes / s
- An ASIC today (2017): 14,000,000 M hashes / s
Summary

- The two generals paradox shows it's impossible to achieve perfect agreement over unreliable networks.

- Byzantine faults can occur when a node deviates from the expected protocol, intentionally or not.

- Extremely difficult to defend against all Byzantine faults; a rogue node might still bring a cluster down.

- Bitcoin is an example of a serious attempt at Byzantine fault tolerance; limitations exist (51% attacks).

Bedtime reading

- The problem of Byzantine generals

- Tangaroa: a Byzantine Fault Tolerant Raft

- How the bitcoin protocol actually works

- DRAM errors in the wild: A Large-Scale Field Study
  [https://research.google.com/pubs/archive/35162.pdf](https://research.google.com/pubs/archive/35162.pdf)

Thanks

Any questions?

Georgios.bitzes@cern.ch
Virtual Machine Images Management in Cloud Environments

Lorena Lobato Pardavila
iCSC, 7th March 2017

About me
- PH-ESE, IT-DB, IT-CM…almost 6 years at CERN.
- DevOps Engineer having fun with BATCH team
  - CSC2013, tCSC2016, iCSC2017
- Interests
  - Computational Neuroscience Systems, Cloud Computing and IT challenges
  - Enjoy traveling a bit too much
- This lecture -> Cloud Images Orchestration

Outline
- First Contact
- Cloud Computing in Distributed Systems
- Cloud Resources Orchestration
- Requirements in Cloud Environments
- Virtual Machine Images: Case Study
- Conclusions

Overview
- First Contact
- Cloud Computing in Distributed Systems
- Cloud Resources Orchestration
- Requirements in Cloud Environments
- Virtual Machine Images: Case Study
- Conclusions
First Contact: Distributed Systems

- Collection of independent computers
- No shared-memory
- Continuous availability
- Easily expandable

First Contact: Grid Computing

- Collection of a large number of computer resources from multiple locations connected together in a network to reach a common goal.
- Sharing computing resources and storage resources
- Many computers acting as a single one

First Contact: Cloud Computing

Wikipedia says that it’s a type of Internet-based computing that provides shared computer processing resources and data to computers and other devices on demand.

Overview

- First Contact
- Cloud Computing in Distributed Systems
- Cloud Resources Orchestration
- Requirements in Cloud Environments
- Virtual Machine Images: Case Study
- Conclusions
Cloud Computing in Distributed Systems

**SERVICE MODELS**

- **Infrastructure as a Service (IaaS)**
  - Compute, Storage, Network, Desktop

- **Platform as a Service (PaaS)**
  - Middleware, Database, Build/Dev/Test

- **Software as a Service (SaaS)**
  - Email, Social Collaboration, Payment and Reconciliation

**Why is this an advantage to the old time of computing models?**
Cloud Computing in Distributed Systems: Virtual Resources

First Contact: Cloud Computing

- Virtual Machines
- Containers
- Images
- ..Of course more!

Overview

- First Contact
- Cloud Computing in Distributed Systems
- Cloud Resources Orchestration
- Requirements in Cloud Environments
- Virtual Machine Image: Case Study
- Conclusions

VIRTUAL MACHINE

CONTAINER

Images

Virtual Resources

....AND WHAT ABOUT THE IMAGES?
Cloud Resources Orchestration

What would it be needed to build a PaaS?

- VM creation
- VM configuration
- Monitoring + Management

Overview

- First Contact
- Cloud Computing in Distributed Systems
- Cloud Resources Orchestration
- Requirements in Cloud Environments
- Virtual Machine Image: Case Study
- Conclusions

Category of cloud computing services that provides a computing platform with a subset of software systems.
Requirements in Cloud Environments

- Software updates
- Diversity of Configuration Management and cloud solutions
- Difficulty to find a “one-size-fits-all” solution

Software updates: To keep Virtual Machines and services updated, bug fixes, etc.

Images need to be up to date!!

- Difficulty to find a “one-size-fits-all” solution: We want everything 😊
- Issue: Don’t have the solution that might fit all of them and there is no so much information about

Source: From rickwell.com, © Rick Weill
Requirements in Cloud Environments

And if we would like wish list it could be nice to have everything automated 😊

Virtual Machines  Adaptability
Keep everything up to the date  Accessibility

Overview

- First Contact
- Scalability in Distributed Systems
- Cloud Resources Orchestration
- Requirements in Cloud Environments
- Virtual Machine Images: Case Study
- Conclusions

Virtual Machine Images: Case Study

- About 20 years ago...
  - Set up machine and network, install OS, install SW, configuring and deploying.
- About 10 years ago...
  - Vmware, VirtualBox, Puppet, Chef, Ansible...everything automatic!
- About 5 years ago...
  - We got Clouds!
  - 1 day configuration
  - 3 hours
  - 30 min

Requirements in Cloud Environments

Overview

• First Contact
• Scalability in Distributed Systems
• Cloud Resources Orchestration
• Requirements in Cloud Environments
• Virtual Machine Images: Case Study
• Conclusions
Virtual Machine Images: Case Study

Why is the management of VM images so important?

Tool for creating identical machine images for multiple platforms from a single source configuration (JSON template)
Virtual Machine Image: Case Study

Overview

- First Contact
- Cloud Computing in Distributed Systems
- Cloud Resources Orchestration
- Requirements in Cloud Environments
- Virtual Machine Images: Case Study
- Conclusions

Conclusions (I)

Cloud
1. Scalability
2. Not direct investment in HW
3. Pay-as-you-use
4. Independent location

PaaS
1. Responsible for the OS
2. Licenses + Software
3. Configuration of the computing resources

Conclusions (II)

Difficulties
1. Variety of Cloud providers
2. Different Clouds
   - Affects the cloud images management
3. Get software updates + new operative system + new operative system versions
Conclusions (III)

Virtual Machine Images Management

• Scalability + Reliability
  • Packer

• System robustness
  • Integration with Rundeck + Puppet + OpenStack + Docker

• Helpful for the automation of the platform

• Helpful for the orchestration of the service
The role of visualization systems is to provide visual representations of datasets that help people carry out tasks more effectively.

Visualization

A Visualization should
1. Save time
2. Have a clear purpose*
3. Include only the relevant content*
4. Encodes data/information appropriately

* from Noel Illinsky, http://complexdiagrams.com/

Course outcomes

The what? Major data types and classifications of them

The why? Why are we visualising at all?

The how? How can we visualize? What archetypes can we use to guide us?

Finally, case study? Given some data, how can we go about visualising it?
A lot of the content for this introduction comes from this book from Prof. Tamara Munzner (UBC, Vancouver, Canada) which I created the illustrations for.

If you’re interested in learning more, it’s a great book to check out from the CERN library, or buy :)
The statistics would lead us to believing that everything is the same.

The role of visualization systems is to provide visual representations of datasets that help people carry out tasks more effectively.

Why visualize?

Analysis framework: Four levels, three questions

Domain situation

Data/task abstraction

Visual encoding/interaction idiom

Algorithm

A Nested Model of Visualization Design and Validation

Analysis framework: Four levels, three questions

• Domain situation
• Domain situation
  - who are the target users?

A Nested Model of Visualization Design and Validation.

Analysis framework: Four levels, three questions

• Domain situation
  - who are the target users?
• Data/Task Abstraction
  - translate from specifics of domain to vocabulary of vis

A Multi-Level Typology of Abstract Visualization Tasks

• Domain situation
  - who are the target users?
• Data/Task Abstraction
  - translate from specifics of domain to vocabulary of vis
    • What is shown? Data abstraction

E. MAGUIRE A. 113 iCSC 2017
Analysis framework: Four levels, three questions

- Domain situation
  - who are the target users?
- Data/Task Abstraction
  - translate from specifics of domain to vocabulary of vis
    - What is shown? Data abstraction
    - Why is the user looking at it? Task abstraction
- Visual Encoding
  - How is it shown?

A Nested Model of Visualization Design and Validation

A Multi-Level Typology of Abstract Visualization Tasks
Analysis framework: Four levels, three questions

• Domain situation
  – who are the target users?

• Data/Task Abstraction
  – translate from specifics of domain to vocabulary of vis
  • What is shown? Data abstraction
  • Why is the user looking at it? Task abstraction

• Visual Encoding
  • How is it shown?
    - visual encoding: how to draw
    - interaction: how to manipulate

• Algorithm
  – efficient computation, layout algorithms etc.

A Nested Model of Visualization Design and Validation

Analysis framework: Four levels, three questions

• Domain situation
  – who are the target users?

• Data/Task Abstraction
  – translate from specifics of domain to vocabulary of vis
  • What is shown? Data abstraction
  • Why is the user looking at it? Task abstraction

• Visual Encoding
  • How is it shown?
    - visual encoding: how to draw
    - interaction: how to manipulate

• Algorithm
  – efficient computation, layout algorithms etc.

A Nested Model of Visualization Design and Validation

Data/Task Abstraction

• What is shown? Data abstraction

• Why is the user looking at it? Task abstraction

Visual Encoding

• How is it shown?
  - visual encoding: how to draw
  - interaction: how to manipulate

Algorithm

A Nested Model of Visualization Design and Validation

Analysis framework: Four levels, three questions

• Domain situation
  – who are the target users?

• Data/Task Abstraction
  – translate from specifics of domain to vocabulary of vis
  • What is shown? Data abstraction
  • Why is the user looking at it? Task abstraction

• Visual Encoding
  • How is it shown?
    - visual encoding: how to draw
    - interaction: how to manipulate

• Algorithm
  – efficient computation, layout algorithms etc.

A Nested Model of Visualization Design and Validation

Analysis framework: Four levels, three questions

• Domain situation
  – who are the target users?

• Data/Task Abstraction
  – translate from specifics of domain to vocabulary of vis
  • What is shown? Data abstraction
  • Why is the user looking at it? Task abstraction

• Visual Encoding
  • How is it shown?
    - visual encoding: how to draw
    - interaction: how to manipulate

• Algorithm
  – efficient computation, layout algorithms etc.

A Nested Model of Visualization Design and Validation

Analysis framework: Four levels, three questions

• Domain situation
  – who are the target users?

• Data/Task Abstraction
  – translate from specifics of domain to vocabulary of vis
  • What is shown? Data abstraction
  • Why is the user looking at it? Task abstraction

• Visual Encoding
  • How is it shown?
    - visual encoding: how to draw
    - interaction: how to manipulate

• Algorithm
  – efficient computation, layout algorithms etc.

A Nested Model of Visualization Design and Validation

Analysis framework: Four levels, three questions

• Domain situation
  – who are the target users?

• Data/Task Abstraction
  – translate from specifics of domain to vocabulary of vis
  • What is shown? Data abstraction
  • Why is the user looking at it? Task abstraction

• Visual Encoding
  • How is it shown?
    - visual encoding: how to draw
    - interaction: how to manipulate

• Algorithm
  – efficient computation, layout algorithms etc.

A Nested Model of Visualization Design and Validation

What are you visualising?

• DATA TYPES
  • TABLES

• DATASET TYPES
  • NETWORKS

• ATTRIBUTE TYPES
  • ORDERED
The branches of data visualization

Information Visualization
Position is given.
Incl. GeoVis

Scientific Visualization
Position is derived.
Also medical visualizations

Why are you visualising this?

Discover
Finding new insights in your data
Implies a level of interactivity to query, compare, correlate etc.
This is typically where one should be careful in how information is presented. An erroneous data encoding could bring about wrong conclusions.

Presenting your results, e.g. for a paper.
Presenting your results, e.g. for a paper
How can you encode information optimally?

<table>
<thead>
<tr>
<th>Arrange</th>
<th>Encode</th>
<th>Manipulate</th>
<th>facet</th>
<th>Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Express</td>
<td>Separation</td>
<td>Change</td>
<td>Filter</td>
<td></td>
</tr>
<tr>
<td>Align</td>
<td>Use</td>
<td>Select</td>
<td>Aggregate</td>
<td></td>
</tr>
</tbody>
</table>

Map:

- Color
- Transparency
- Position, size, angle, curvature, etc.

Note: Bear in mind that this was a non-interactive figure in a paper.
How can you encode information optimally?

- **Arrange**
  - Express
  - Separate
  - Order
  - Align
  - Use

- **Map**
  - from qualitive and quantitative attribute
  - Color
  - Hue
  - Saturation
  - Transparency
  - Region, Texture, Shape, ...
  - Motion
  - Distance, Rate, Frequency, ...
  - Position, Size, Angle, Curvature, ...

**Manipulate**
- Change
- Select
- Navigate

**Facet**
- Justapose
- Partion
- Superimpose

**Reduce**
- Filter
- Aggregate
- Embed

---

How can you encode information optimally?

<table>
<thead>
<tr>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
<th>APR</th>
<th>MAY</th>
<th>JUN</th>
<th>JUL</th>
<th>AUG</th>
<th>SEP</th>
<th>OCT</th>
<th>NOV</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

---

How can you encode information optimally?

<table>
<thead>
<tr>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
<th>APR</th>
<th>MAY</th>
<th>JUN</th>
<th>JUL</th>
<th>AUG</th>
<th>SEP</th>
<th>OCT</th>
<th>NOV</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

---

How can you encode information optimally?

<table>
<thead>
<tr>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
<th>APR</th>
<th>MAY</th>
<th>JUN</th>
<th>JUL</th>
<th>AUG</th>
<th>SEP</th>
<th>OCT</th>
<th>NOV</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

---

How can you encode information optimally?

<table>
<thead>
<tr>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
<th>APR</th>
<th>MAY</th>
<th>JUN</th>
<th>JUL</th>
<th>AUG</th>
<th>SEP</th>
<th>OCT</th>
<th>NOV</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

---

How can you encode information optimally?

<table>
<thead>
<tr>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
<th>APR</th>
<th>MAY</th>
<th>JUN</th>
<th>JUL</th>
<th>AUG</th>
<th>SEP</th>
<th>OCT</th>
<th>NOV</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

---

How can you encode information optimally?

<table>
<thead>
<tr>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
<th>APR</th>
<th>MAY</th>
<th>JUN</th>
<th>JUL</th>
<th>AUG</th>
<th>SEP</th>
<th>OCT</th>
<th>NOV</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>
How can you encode information optimally?
How can you encode information optimally?

And that's just a really simple low dimensional example

Moreover, all of these visualizations encode the information, but the decode error (interpreting, comparing...) for each is different

So, why?

Our perception system does not behave linearly.

Some stimuli are perceived less or more than intended.

Steven's Psychophysical Power Law: $S = I^N$

Stevens, 1975
We have to be careful when mapping data to the visual world

Some visual channels are more effective for some data types over others.
Natural Mappings

We have to be careful when mapping data to the visual world. Some visual channels are more effective for some data types over others. Some data has a natural mapping that our brains expect given certain types of data. There are many intricacies of the visual system that must be considered.

The pop-out effect
We pre-attentively process a scene, and some visual elements stand out more than others.

- Parallel processing on many individual channels
  - speed independent of distractor count
  - speed depends on channel and amount of difference from distractors
- Serial search for (almost all) combinations
  - speed depends on number of distractors

- Parallel processing on many individual channels
  - speed independent of distractor count
  - speed depends on channel and amount of difference from distractors
- Serial search for (almost all) combinations
  - speed depends on number of distractors
The pop-out effect
We pre-attentively process a scene, and some visual elements stand out more than others.

- Parallel processing on many individual channels
  - speed independent of distractor count
  - speed depends on channel and amount of difference from distractors
- Serial search for (almost all) combinations
  - speed depends on number of distractors

The pop-out effect
We pre-attentively process a scene, and some visual elements stand out more than others.

- Parallel processing on many individual channels
  - speed independent of distractor count
  - speed depends on channel and amount of difference from distractors
- Serial search for (almost all) combinations
  - speed depends on number of distractors
Not all exhibit the pop-out effect!
Parallel line pairs do not pop out from tilted pairs.
And not all visual channels pop out as quickly as other. E.g. colour is always on top.
Known Target Search

Colour Variety

Unknown Target Search

Grouped

Random

Subitizing

Grouped

Random

Which grid has more colours?

7 8

B) Subitizing (how many colours?)

How Capacity Limits of Attention Influence Information Visualization Effectiveness.

Haroz S. and Whitney D., IEEE TVCG 2012

A) Known and Unknown Target Search

C) Response Time and Accuracy Results

E. MAGUIRE

iCSC 2017
A. Law of Closure  
B. Law of Similarity  
C. Law of Proximity  
D. Law of Connectedness  
E. Law of Symmetry  
F. Law of Good Continuation  
G. Contour Saliency  
H. Law of Common Fate  
I. Law of Past Experience  
J. Law of Pragnanz  
K. Figure/Ground

**Integral/Separable Dimensions**

- **Integral Dimensions**:  
  - Width and Height are integral dimensions.  
  - **Example**: A and B have the same width. However, B and C are perceived more alike even though they are different widths and heights.

- **Separable Dimensions**:  
  - Colour and Height are separable dimensions.  
  - **Example**: A and B have the same colour and are perceived more similar. Colour and Height are separable dimensions.
Integral/Separable Dimensions

We have to be careful when mapping data to the visual world

Some visual channels are more effective for some data types over others.

Some data has a natural mapping that our brains expect given certain types of data.

There are many visual tricks that can be observed due to how the visual system works.

We don’t see in 3D, and we have difficulties interpreting information on the Z-axis.

2D always wins…

Our visual system is not good at interpreting information on the z-axis.

*3D is normally only used for exploration of inherently 3D information, such as medical imaging data…
2D always wins…

We have to be careful when mapping data to the visual world

Some visual channels are more effective for some data types over others.
Some data has a natural mapping that our brains expect given certain types of data.
There are many visual tricks that can be observed due to how the visual system works.
We don’t see in 3D, and we have difficulties interpreting information on the Z-axis.

Colour

Additionally, colour is not equally binned in reality. We perceive colours differently due to an increased sensitivity to the yellow part of the spectrum.

Luminosity is also not stable across the colours, meaning some colours will pop out more than others, and not always intentionally.

The problem is that a smooth step in a value does not equate to a smooth colour transition.

https://mycarta.wordpress.com/2012/10/06/the-rainbow-is-dead-long-live-the-rainbow-part-3/
Luminosity is also not stable across the colours, meaning some colours will pop out more than others... and not always intentionally.

https://mycarta.wordpress.com/2012/10/06/the-rainbow-is-dead-long-live-the-rainbow-part-3/

And how we perceive changes in hue is also very different.

Gregory compared the wavelength of light with the smallest observable difference in hue (expressed as wavelength difference)

Is this a good visualization?

But, grayscale would be just as useful and less visually distracting.
Is there a colour palette for scientific visualization that works?

HSL linear L rainbow palette

But there are some in CMS that are already moving away from the rainbow

https://mycarta.wordpress.com/2012/10/06/the-rainbow-is-dead-long-live-the-rainbow-part-3/


Is there a colour palette for scientific visualization that works?

There are also lots of default colour maps that can be applied to particular data types.

Semantic relevance
Or just consistency

When there are many colours for example, we find it difficult to remember abstract associations.

https://mycarta.wordpress.com/2012/10/06/the-rainbow-is-dead-long-live-the-rainbow-part-3/


http://colorbrewer2.org/
What are semantically resonant colours?

Semantic colouring is a good idea in theory, but there are limited areas where this really works.

But, if you are going to use colour, have a consistent colour mapping. That way, the decoding time is less.

Saving time...

In general, CMS & other experiments are pretty good at this, but there are some examples...
And that was just to represent a low number of dimensions.

What happens when we have a high number of dimensions?

### Multidimensional Visualization

Scatter Plot Matrices

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Weight</th>
<th>Chol</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1.76</td>
<td>63</td>
<td>4.5</td>
</tr>
<tr>
<td>Mike</td>
<td>1.79</td>
<td>70</td>
<td>4.15</td>
</tr>
<tr>
<td>Jim</td>
<td>1.61</td>
<td>60</td>
<td>6.7</td>
</tr>
<tr>
<td>Francois</td>
<td>1.84</td>
<td>90</td>
<td>5.03</td>
</tr>
</tbody>
</table>

Temperature (Celsius), Wind-direction (°), Wind Speed (km/h), Location: Paris
### Multidimensional Visualization

#### Scatter Plot Matrices

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Weight</th>
<th>Chol</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1.76</td>
<td>63</td>
<td>4.5</td>
</tr>
<tr>
<td>Mike</td>
<td>1.79</td>
<td>70</td>
<td>4.15</td>
</tr>
<tr>
<td>Jim</td>
<td>1.61</td>
<td>60</td>
<td>6.7</td>
</tr>
<tr>
<td>Francois</td>
<td>1.84</td>
<td>90</td>
<td>5.03</td>
</tr>
</tbody>
</table>

---

### Multidimensional Visualization

#### Scatter Plot Matrices

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Weight</th>
<th>Chol</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1.76</td>
<td>63</td>
<td>4.5</td>
</tr>
<tr>
<td>Mike</td>
<td>1.79</td>
<td>70</td>
<td>4.15</td>
</tr>
<tr>
<td>Jim</td>
<td>1.61</td>
<td>60</td>
<td>6.7</td>
</tr>
<tr>
<td>Francois</td>
<td>1.84</td>
<td>90</td>
<td>5.03</td>
</tr>
</tbody>
</table>
Multidimensional Visualization

Linked Plots


Multidimensional Visualization

When one visualization won’t cut it...

Multidimensional Visualization

With dc.js, crossfilter, and d3.js

Cite Summary

Multidimensional Visualization

With dc.js, crossfilter, and d3.js

E. MAGUIRE

A. 141

iCSC 2017
Multidimensional Visualization
With dc.js, crossfilter, and d3.js

My Tutorial on Creating Dashboard Visualizations
https://thor-project.github.io/dashboard-tutorial/
Multidimensional Visualization

Parallel Coordinate Plots

Positive Correlation
Negative (inverse) Correlation
No Correlation

Positive Correlation
Negative (inverse) Correlation
No Correlation

Multidimensional Visualization

Parallel Coordinate Plots

Lets take an example where we have many variables to display... Each user is represented by a circle.

2 Dimensions
3 Dimensions
Size indicates number of logins per day

4 Dimensions
Color indicates users' department

5 Dimensions
Transparency indicates consistency in logins

As we get to higher levels of dimensions, we'll have problems. Our choice of visual encoding will affect the visual availability of each dimension to the user.
Parallel coordinates are a visualization technique employed when a large number of dimensions need to be displayed (often without a temporal element) and where each of those dimensions can be equally important in the decision making process.

In the scatter plots here, it’s easy to see correlation between downloads and uploads, but with the other dimensions that’s difficult.
Parallel Coordinate Plots

We can keep adding more parallel lines, and comfortably have around 20 dimensions for many users displayed at once.

Multidimensional Visualization

Glyphs

- static item aggregation
- **task**: find distribution
- **data**: table
- derived data
  - 4 quantitative attributes
  - median: central line
  - lower and upper quartile: boxes
  - lower upper fences: whiskers
  - outliers beyond fence cutoffs explicitly shown

Figure 1: From left to right: box plots vase plots violin plot and bean plot. Within each plots the distributions from left to right are: standard normal, right skewed, leptokurtic, and bimodal. A normal kernel and bandwidth of 0.05 are used in all plots for all groups.

A more sophisticated display is the sectioned density plot which uses both colour and space to stack a density estimate into a smaller area hopefully without losing any information. The sectioned density plot is similar in spirit to horizon graphs for time series which have been found to be just as readable as regular line graphs despite taking up much less space. The density strips of Jackson provide a similar compact display that uses colour instead of width to display density. These methods are shown in Figure 5.
What about topological data?

Trees

Treemaps Partitioning

- split by neighbourhood
- then by type
- colour by price

- neighbourhood patterns
  - where it’s expensive
  - where you pay much more for detached type


Treemaps Partitioning

- switch order of splits
  - type then neighbourhood
- switch colour
  - by price variation
- type patterns
  - within specific type, which neighbourhoods are inconsistent

Validation & User Testing

Validating your visualisation...

Domain situation
Observe target users using existing tools
Visual encoding/interaction idiom
Justify design with respect to alternatives
Algorithm
Measure system time/memory
Analyze computational complexity

Data/task abstraction
Analyze results qualitatively
Measure human time with lab experiment (lab study)
Observe target users after deployment (field study)
Measure adoption

Validating your visualisation...

The elephant in the room… scaling up

“Visualization can surprise you, but doesn’t scale well. Modelling scales well, but can’t surprise you.”

Hadley Wickham

The elephant in the room… scaling up

One of the biggest challenges in HEP, Biology, chemistry, and business is scale.

Our screens have a limited number of pixels. And our data is often much larger.

You can do two things to get around this.
Solutions

Reduce the problem space
Try and get users to focus queries as soon as possible to reduce the amount of data to be visualised.

Provide ways to aggregate information into meaningful overview visualisations.

GPU usage and WebGL
As done in imMens from Jeffrey Heer’s lab in Washington. Renders billions of data points at 50 fps in the browser.
Not yet compatible across all browsers, and not everyone has a dedicated GPU.

Further Links

Tutorial on D3
http://antarctic-design.co.uk/biovis-workshop15/

Tutorial on Dashboard Visualizations
https://thor-project.github.io/dashboard-tutorial/

Visualization Survey Sites
Set Visualization - http://www.cvast.tuwien.ac.at/SetViz
Time Series Visualization - http://survey.timeviz.net/
Parallel Coordinates Visualization
Periodic Table of Visualizations
Data Vis Catalogue


Questions
@antarcticdesign
eamonnmag@gmail.com
For more information on the CERN School of Computing:

http://cern.ch/csc