



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXPLORANDO EL POTENCIAL DE LAS REDES NEURONALES DE GRAFOS
PARA EXTRAER INFORMACIÓN DE BGP

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN COMPUTACIÓN

VALENTINA FRANCISCA ESTEBAN LAGOS

PROFESORA GUÍA:
IVANA BACHMANN ESPINOZA

PROFESOR CO-GUÍA:
SEBASTIÁN FERRADA

MIEMBROS DE LA COMISIÓN:
NOMBRE UNO
NOMBRE DOS
NOMBRE TRES

SANTIAGO DE CHILE
2025

dedicatoria.

Índice

1. Introducción	1
1.1. Motivación	2
1.2. Hipótesis	2
1.3. Objetivos	2
1.3.1. Objetivo general	3
1.3.2. Objetivos específicos	3
1.4. Metodología	3
1.5. Contribuciones	4
1.6. Estructura del trabajo	4
2. Infraestructura de Internet	5
3. Border Gateway protocol (BGP)	6
3.1. OPEN message	7
3.2. UPDATE message	7
3.3. KEEPALIVE message	13
3.4. NOTIFICATION message	14
3.5. BGP Communities	16
4. AS Inference	17
4.1.1. Gao	18
4.1.2. Lu Rank	19
4.1.3. BGP2Vec	20
5. Redes Neuronales	24
5.1. Grafos	24
5.2. Inteligencia Artificial	25
5.3. Redes Neuronales	26
5.3.1. Funciones de Activación	28
5.3.2. Función de Pérdida	28
5.3.3. Tipos de Redes Neuronales	28
5.3.4. Redes Neuronales Feed Forward (FFNN)	28
5.3.5. Redes Neuronales Recursivas (RNN)	29
5.3.6. Redes Neuronales Convolucionales (CNN)	29
5.3.7. Graph Neuronla Network (GNN)	30
5.3.7.1. Message Passing Neural Networks (MPNN)	32

5.3.7.2. Graph Convolution Network (GCN)	33
5.3.7.3. Graph Attention Network (GAT)	34
5.3.7.4. GraphSAGE (SAmple and aggreGatE)	35
5.3.8. Procesamiento de Datos	37
5.3.9. Entrenamiento	37
5.3.9.1. Optimización	37
5.3.9.2. Sampling (Muestreo)	43
5.3.9.3. Regularización	48
5.3.10. Evaluación	50
5.3.10.1. Metricas de evaluación	51
6. Internat Data collection	55
6.1. CAIDA	55
6.2. RouteViews Project	56
6.3. RIPE NCC RIS	57
6.4. The peering DB	57
6.5. Hurricane Electric Internet Services	58
7. Experimentos	59
7.1. Benchmark	59
7.2. Datos	59
7.3. modelos	59
7.4. Dataset de Evaluación	59
8. Experimentos	60
8.1. Dataset de Validación	60
8.2. Experimento 1:	60
8.3. Experimento 2:	61
8.4. Experimento 3:	62
8.5. Experimento 4:	62
9. Resultados y Analisis	67
Bibliografía	69
Anexo A. Vocabulario	72
Anexo B. titulo anexo 1	72
Anexo C. Cosas Extras	76

Índice de Tablas

Índice de Ilustraciones

Figura 4.1: ???	21
Figura 4.2: ???	21
Figura 5.1: Representación de un grafo no dirigido de 6 nodos numerados.	24
Figura 5.2: jerarquía conceptual entre Inteligencia Artificial, Machine Learning y Deep Learning.	26
Figura 5.3: Arquitectura básica de una Red Neuronal Fully Connected de una capa. . .	27
Figura 5.4: Estructura general de un perceptrón.	28
Figura 5.5: Arquitectura básica de las Redes Neuronales Recurrentes.	29
Figura 5.6: Estructura de un modelo de Convolucional con tres capas.	30
Figura 5.7: Pipeline básico de una arquitectura GNN.	32
Figura 5.8: Flujo de Message passing para un nodo del grafo.	33
Figura 5.9: Pipeline del mecanismo de cálculo de las ponderaciones para una GAT.	35
Figura 5.10: TODO.	36
Figura 5.11: Inductive y transductive settings para entrenar y testear un modelo GNN. 37	37
Figura 5.12: Visualización esquemática del proceso de entrenamiento en un modelo de Red Neuronal.	38
Figura 5.13: visualizaacion optimixacion del aLgoritmo de Gradient Descent aolicada a una funcion convexa.	39
Figura 5.14: Batch Gradient Descent.	40
Figura 5.15: Batch Gradient Descent.	40
Figura 5.16: Stochastic Gradient Descent.	41
Figura 5.17: Stochastic Gradient Descent.	41
Figura 5.18: Batch Gradient Descent.	41
Figura 5.19: Tipos de Entrenamineto.	41
Figura 5.20: Descenso del Gradiente con Momentum:	42
Figura 5.21: visualizaacion learning Rate grande y chico.	43
Figura 5.22: Sampling.	44
Figura 5.23: Cluster sampling.	45
Figura 5.24: Cluster sampling.	46

Figura 5.25: Explciacion FIXME:	48
Figura 5.26: Evaluation.	49
Figura 5.27: Dropout.	50
Figura 5.28: Matriz de confusión.	51
Figura 5.29: ROC Curve.	53
Figura 8.1: Resultados.	61
Figura 8.2: Resultados.	61
Figura 8.3: Resultados.	62
Figura 8.4: Resultados.	62
Figura 9.1: El gatito más bello del mundo.	68

Capítulo 1

Introducción

En la sociedad actual el Internet juega un papel esencial en la vida cotidiana, facilitando la comunicación, la colaboración y el intercambio de información. En los últimos años, su uso y desarrollo ha crecido exponencialmente, convirtiéndose en una herramienta indispensable. Esta creciente interconexión global ha hecho que las redes sean fundamentales para el funcionamiento de la sociedad moderna. En este contexto, es crucial entender cómo está formado el Internet, ya que conocer su funcionamiento y estructura permite preservar su integridad y eficiencia. Además, garantizar un funcionamiento continuo y óptimo.\

El Internet está conformado por miles de Sistemas Autónomos (SA) interconectados entre sí. Cada SA consiste en un conjunto de IPs que comparten un mismo protocolo de enrutamiento y están administradas por una misma entidad, como proveedores de servicios de Internet (ISP), empresas comerciales, universidades, entre otros. A cada uno de estos Sistemas Autónomos se le asigna un número y prefijos de direcciones IP, los cuales anuncia a sus vecinos a través del Border Gateway Protocol (BGP). BGP es un protocolo dinámico de enrutamiento externo en el que los SA anuncian sus tablas de ruteo y cambios en sus AS-Paths para alcanzar direcciones IP específicas. De esta manera, cada SA recibe estos anuncios de todos sus vecinos BGP y toma decisiones sobre la mejor forma de direccionar sus paquetes. \

Dentro del grafo de Sistemas Autónomos que conforma Internet, el camino que un paquete recorre de un nodo a otro no suele ser el más corto debido a los acuerdos comerciales que cada SA, como entidad independiente, establece con sus vecinos. Estos acuerdos se clasifican en tres tipos de relaciones: 1) Provider-to-customer (P2C), en el cual el cliente paga al SA proveedor para que este enlace permita el tráfico de sus paquetes hacia el resto de Internet. 2) Peer-to-peer (P2P), donde los SA intercambian tráfico entre sí y con sus clientes, pero no con sus proveedores u otros pares. 3) Sibling-to-sibling (S2S), cuando dos

SA pertenecen al mismo dominio. Gao [??] propuso reglas para modelar estas relaciones entre SA, que reflejan cómo suelen configurarse en BGP, lo que permite inferir las posibles rutas seleccionadas por este protocolo. Sin embargo, estas soluciones se basan principalmente en el cálculo de heurísticas. Por otro lado, estudios más recientes como el de Shapira y Shavitt[1] proponen técnicas de Deep Learning, creando representaciones de los SA que luego son utilizadas en una Red Neuronal.\

Aquí entra en juego un nuevo enfoque que podría cambiar nuestra forma de analizar datos: el uso de Redes Neuronales de Grafos (GNNs). Las GNNs están diseñadas específicamente para trabajar con datos organizados en forma de grafos. Al ser Redes Neuronales, tienen la capacidad de encontrar representaciones efectivas de la información y descubrir patrones en datos estructurados de esta manera. A diferencia de las Redes Neuronales convencionales, las GNNs pueden aprovechar la información de los nodos vecinos, lo que les permite entender mejor la estructura del grafo y realizar análisis más detallados.\

Así nace esta tesis, con el objetivo de explorar el comportamiento de las Redes Neuronales de Grafos (GNNs) utilizando datos de BGP para representar Internet y las relaciones entre los Sistemas Autónomos que lo componen. Esta área es aún poco explorada y presenta varias dificultades, principalmente debido a la necesidad de aplicar conocimiento tanto en redes como en Deep Learning. En particular, obtener una representación precisa de la topología de Internet requiere comprender a fondo el funcionamiento del protocolo BGP y saber cómo y dónde obtener datos, que no siempre están disponibles públicamente. Así como también tener un conocimiento de teoría de grafos y Deep Learning necesario al momento de la implementación de un modelo de GNNs y diferentes técnicas para la tarea requerida.

1.1 Motivación

(está copiado el problema, es similar pero revisar de cambiar la forma en cómo se plantea)

1.2 Hipótesis

Las Redes Neuronales de Grafos (GNNs) pueden ofrecer un rendimiento superior en comparación con las metodologías del estado del arte [1] para la inferencia del tipo de relación entre Sistemas Autónomos.

1.3 Objetivos

1.3.1 Objetivo general

El objetivo principal de este estudio es evaluar diversas arquitecturas de Redes Neuronales de Grafos (GNNs) para determinar su viabilidad en la inferencia del tipo de relación de tráfico entre dos Sistemas Autónomos. Esto se logrará mediante el análisis de características específicas de cada Sistema Autónomo, la información de actualizaciones BGP, la topología y los cambios en esta.

1.3.2 Objetivos específicos

1. Obtención de datos: Recopilar datos de fuentes confiables como

que correspondan a Sistemas Autónomos representativos de la Red de Internet. Esto implica obtener datos sobre nodos, características y relaciones entre ellos. Asimismo, obtener información relevante sobre flujos de paquetes BGP.

1. Preparación de datos: Mejorar la calidad de los datos mediante el uso de técnicas de normalización, conversión de atributos categóricos a numéricos, manejo de desequilibrio de clases, entre otros.

Además, construir el grafo y definir cómo se proporcionarán los datos de entrada a nuestros modelos GNNs.

1. Diseño e implementación de modelos: Diseñar e implementar modelos GNN y framework específicos que permita la inferencia del tipo de relación que dos Sistemas Autónomos comparten.
2. Evaluación de performance: Comparar el desempeño de diferentes arquitecturas de GNNs en las inferencias, identificando los parámetros de mayor relevancia.
3. Análisis de resultados: Comprender los resultados obtenidos mediante el estudio y la comparación con los valores esperados y estado del arte [1].

1.4 Metodología

El plan de trabajo que se espera llevar a cabo durante esta investigación consta de cuatro etapas:

Investigación y familiarización En esta primera etapa, se llevará a cabo la lectura de artículos académicos relacionados con el uso de GNNs, además de artículos relevantes en la representación de datos de internet, con el objetivo de adquirir conocimiento sobre el problema en cuestión. Al mismo tiempo, se realizará un estudio detallado de datasets representativos de internet y, más importante aún, de la topología de BGP, junto con actualizaciones de estos e información adicional que se puede obtener tan-

to de sistemas autónomos como de los paquetes que intercambian. En paralelo a la investigación, se procederá al desarrollo de modelos básicos de GNNs con el propósito de familiarizarse con las herramientas que se utilizarán a lo largo del proyecto.

Preparación de datos Una vez se tenga información sobre la topología BGP, los Sistemas Autónomos que la componen y los tipos de relaciones de entre ellos, se procederá a convertir los datos a la representación de entrada que nuestro modelo recibirá. Esto también implica el uso de diversas técnicas destinadas a mejorar la calidad de los datos. El enfoque de esta etapa dependerá del estado inicial de los datos, lo que podría implicar acciones como la limpieza de datos, normalización y reducción de la variabilidad, entre otros procesos que se consideren necesarios.

Construcción de modelos y entrenamiento Una vez finalizada la investigación y la familiarización con el problema y las herramientas pertinentes, se dará inicio a la implementación de diversos frameworks y metodologías, utilizando diferentes modelos de GNNs con el conjunto de datos. Posteriormente, se procederá a entrenar los modelos y a ajustar los hiperparámetros o realizar cambios según sea necesario. Se realizará un seguimiento de los resultados, comparándolos con los hallazgos de los artículos académicos previamente revisados. Esto permitirá un proceso de mejora continua, aprendizaje y adaptación en la creación de estos modelos.

Análisis de resultados Una vez terminada la construcción de los modelos, se procederá a analizar los resultados obtenidos para finalmente empezar a escribir el informe de esta tesis.

1.5 Contribuciones

1.6 Estructura del trabajo

la tesis está organizada de la siguiente manera:

- capítulo 2 blabla
- capítulo 3 blablabla
- capítulo 4 no existe todavía

Capítulo 2

Infraestructura de Internet

El internet consiste en una gran colección de host interconectados. Este se puede dividir en Sistemas administrativos los cuales poseen uno o varios Sistemas Autónomos. Dominios administrativos pueden ser desde college campuses, corporate networks , Large ISP (Internet Service Providers). Cada Sistema Autónomo es representado por un número de 16 bits llamado ASN (Autonomous System Number), lo que hace que existan 65536 ASN posibles, de los cuales no todos están asignados a dominios administrativos y no todos los asignados son usados. Muchos ISPs poseen más de un solo ASes.

Cada Sistema Autónomo tiene sus propios routers y routing policies a través del cual se comunica con otros Sistemas Autónomos para intercambiar tráfico.[Informe/capitulos/1-Introduccion.pdf](#)

Los Internet Exchange Points (IXPs) son puntos de interconexión entre redes, funcionan como un medio compartido. Tipicamente, consiste en un switch (capa 2) de alta velocidad y capacidad que permite la interconexión de routers pertenecientes a distintos Sistemas Autónomos. Esto facilita el establecimiento de relaciones de peering entre los Sistemas Autónomos, permitiendo el intercambio de tráfico de manera eficiente y a bajo costo. Algunos IXPs existentes en Chile son PIT Chile y NAP Chile, quienes establecen conexiones. En Chile existe una normativa que regula la interconexión nacional según la cual, todos los ISP que operen deben estar interconectados entre sí. La interconexión nacional se logra conectándose a NAP Chile.

Cada Sistema Autónomo tiene la responsabilidad de

Capítulo 3

Border Gateway protocol (BGP)

Un Sistema Autonomo es un «Conjunto de routers bajo una sola administración técnica» [RFC1771]. Estos a traves de sesiones BGP intercambian rutas conocidas como AS Paths. Utilizando estas rutas podemos mapear la topología de Internet. Sin embargo BGP utiliza la «mejor» ruta entre ASNs, a pesar de que pueden haber más de una ruta entre dos ASes. Para este problema se usan multiples views (viewpoints) Aun asi no se tiene una topología completa.

Border Gateway Protocol is an inter-autonomous system routing protocol designed for TCP/IP Internets.[2]

- tipo: Path Vector
- Puerto: TCP 179
- RFC 4271
- The initial data flow is the portion of the BGP routing table that is allowed by the export policy, called the Adj-RIB-Out.
- Incremental updates are sent as the routing table changes.
- KEEPALIVE messages must be sent periodically to ensure that the connection is still alive.
- NOTIFICATION messages are sent in response to error or special conditions.(if connection encounters an error conditions or NOTIFICATION message is sent and the connection is closed)

3.1 OPEN message

Luego de establecida la conexión BGP, el primer mensaje que se envía por ambos lados de la conexión es un mensaje OPEN. Si es aceptado el mensaje OPEN, se envía un mensaje KEEPALIVE confirmando la conexión.

El mensaje OPEN tiene un tamaño mínimo de 29 octetos incluido el header, además de un header contiene los siguientes campos:

- **Version:** Indica la versión del protocolo BGP.
- **My Autonomous System:** Indica el número del sistema autónomo del emisor.
- **Hold Time:** Indica el número de segundos que propone el emisor para el valor del Hold Timer. Este valor indica el número máximo de segundos que puede pasar entre la recepción de un mensaje KEEPALIVE y/o UPDATE del emisor.
- **BGP Identifier:** Identificador BGP del emisor.
- **Optional Parameters Length:** Indica el largo de los parámetros opcionales. Si este valor es 0, no hay parámetros opcionales.
- **Optional Parameters:** Contiene una lista de los parámetros opcionales.

3.2 UPDATE message

UPDATE messages are used to transfer routing information between BGP peers. The information in the UPDATE message can be used to construct a graph that describes the relationships of the various Autonomous Systems. By applying rules to be discussed, routing

information loops and some other anomalies may be detected and removed from inter-AS routing.

An UPDATE message is used to advertise feasible routes that share common path attributes to a peer, or to withdraw multiple unfeasible routes from service (see 3.1). An UPDATE message MAY simultaneously advertise a feasible route and withdraw multiple unfeasible routes from service. The UPDATE message always includes the fixed-size BGP header, and also includes the other fields, as shown below (note, some of the shown fields may not be present in every UPDATE message): Withdrawn Routes Length:

This 2-octet unsigned integer indicates the total length of the Withdrawn Routes field in octets. Its value allows the length of the Network Layer Reachability Information field to be determined, as specified below.

A value of 0 indicates that no routes are being withdrawn from service, and that the WITHDRAWN ROUTES field is not present in this UPDATE message.

Withdrawn Routes:

This is a variable-length field that contains a list of IP address prefixes for the routes that are being withdrawn from service. Each IP address prefix is encoded as a 2-tuple of the he use and the meaning of these fields are as follows:

a) Length:

The Length field indicates the length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with prefix, itself, of zero octets).

b) Prefix:

The Prefix field contains an IP address prefix, followed by the minimum number of trailing bits needed to make the end of the field fall on an octet boundary. Note that the value of trailing bits is irrelevant.

Total Path Attribute Length:

This 2-octet unsigned integer indicates the total length of the Path Attributes field in octets. Its value allows the length of the Network Layer Reachability field to be determined as specified below.

A value of 0 indicates that neither the Network Layer Reachability Information field nor the Path Attribute field is present in this UPDATE message.

Path Attributes:

A variable-length sequence of path attributes is present in every UPDATE message, except for an UPDATE message that carries only the withdrawn routes. Each path attribute is a triple length.

Attribute Type is a two-octet field that consists of the Attribute Flags octet, followed by the Attribute Type Code octet.

0	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	Attr.
Flags		Attr.	Type	Code	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

The high-order bit (bit 0) of the Attribute Flags octet is the Optional bit. It defines whether the attribute is optional (if set to 1) or well-known (if set to 0).

Rekhter, et al. Standards Track [Page 16]

RFC 4271 BGP-4 January 2006

The second high-order bit (bit 1) of the Attribute Flags octet is the Transitive bit. It defines whether an optional attribute is transitive (if set to 1) or non-transitive (if set to 0).

For well-known attributes, the Transitive bit MUST be set to 1. (See Section 5 for a discussion of transitive attributes.)

The third high-order bit (bit 2) of the Attribute Flags octet is the Partial bit. It defines whether the information contained in the optional transitive attribute is partial (if set to 1) or complete (if set to 0). For well-known attributes and for optional non-transitive attributes, the Partial bit MUST be set to 0.

The fourth high-order bit (bit 3) of the Attribute Flags octet is the Extended Length bit. It defines whether the Attribute Length is one octet (if set to 0) or two octets (if set to 1).

The lower-order four bits of the Attribute Flags octet are unused. They MUST be zero when sent and MUST be ignored when received.

The Attribute Type Code octet contains the Attribute Type Code. Currently defined Attribute Type Codes are discussed in Section 5.

If the Extended Length bit of the Attribute Flags octet is set to 0, the third octet of the Path Attribute contains the length of the attribute data in octets.

If the Extended Length bit of the Attribute Flags octet is set to 1, the third and fourth octets of the path attribute contain the length of the attribute data in octets.

Rekhter, et al. Standards Track [Page 17]

The remaining octets of the Path Attribute represent the attribute value and are interpreted according to the Attribute Flags and the Attribute Type Code. The supported Attribute Type Codes, and their attribute values and uses are as follows:

a) ORIGIN (Type Code 1):

ORIGIN is a well-known mandatory attribute that defines the origin of the path information. The data octet can assume the following values:

Value Meaning

0 IGP - Network Layer Reachability Information is interior to the originating AS

1 EGP - Network Layer Reachability Information learned via the EGP protocol [RFC904]

2 INCOMPLETE - Network Layer Reachability Information learned by some other means

Usage of this attribute is defined in 5.1.1.

b) AS_PATH (Type Code 2):

AS_PATH is a well-known mandatory attribute that is composed of a sequence of AS path segments. Each AS path segment is length, path segment value>.

The path segment type is a 1-octet length field with the following values defined:

Value Segment Type

1 AS_SET: unordered set of ASes a route in the UPDATE message has traversed

2 AS_SEQUENCE: ordered set of ASes a route in the UPDATE message has traversed

The path segment length is a 1-octet length field, containing the number of ASes (not the number of octets) in the path segment value field.

The path segment value field contains one or more AS numbers, each encoded as a 2-octet length field.

Rekhter, et al. Standards Track [Page 18]

RFC 4271 BGP-4 January 2006

Usage of this attribute is defined in 5.1.2.

c) NEXT_HOP (Type Code 3):

This is a well-known mandatory attribute that defines the (unicast) IP address of the router that SHOULD be used as the next hop to the destinations listed in the Network Layer Reachability Information field of the UPDATE message.

Usage of this attribute is defined in 5.1.3.

d) MULTI_EXIT_DISC (Type Code 4):

This is an optional non-transitive attribute that is a four-octet unsigned integer. The value of this attribute MAY be used by a BGP speaker's Decision Process to discriminate among multiple entry points to a neighboring autonomous system.

Usage of this attribute is defined in 5.1.4.

e) LOCAL_PREF (Type Code 5):

LOCAL_PREF is a well-known attribute that is a four-octet unsigned integer. A BGP speaker uses it to inform its other internal peers of the advertising speaker's degree of preference for an advertised route.

Usage of this attribute is defined in 5.1.5.

f) ATOMIC_AGGREGATE (Type Code 6)

ATOMIC_AGGREGATE is a well-known discretionary attribute of length 0.

Usage of this attribute is defined in 5.1.6.

g) AGGREGATOR (Type Code 7)

AGGREGATOR is an optional transitive attribute of length 6. The attribute contains the last AS number that formed the aggregate route (encoded as 2 octets), followed by the IP address of the BGP speaker that formed the aggregate route (encoded as 4 octets). This SHOULD be the same address as the one used for the BGP Identifier of the speaker.

Usage of this attribute is defined in 5.1.7.

Rekhter, et al. Standards Track [Page 19]

RFC 4271 BGP-4 January 2006

Network Layer Reachability Information:

This variable length field contains a list of IP address prefixes. The length, in octets, of the Network Layer Reachability Information is not encoded explicitly, but can be calculated as:

- UPDATE message Length - 23 - Total Path Attributes Length
- Withdrawn Routes Length

where UPDATE message Length is the value encoded in the fixed- size BGP header, Total Path Attribute Length, and Withdrawn Routes Length are the values encoded in the variable part of the UPDATE message, and 23 is a combined length of the fixed- size BGP header, the Total Path Attribute Length field, and the Withdrawn Routes Length field.

Reachability information is encoded as one or more 2-tuples of

+—————+ | Length (1 octet) | +—————+ | Prefix (variable)
| +—————+ |

The use and the meaning of these fields are as follows:

- a) Length:

The Length field indicates the length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with prefix, itself, of zero octets).

b) Prefix:

The Prefix field contains an IP address prefix, followed by enough trailing bits to make the end of the field fall on an octet boundary. Note that the value of the trailing bits is irrelevant.

The minimum length of the UPDATE message is 23 octets – 19 octets for the fixed header + 2 octets for the Withdrawn Routes Length + 2 octets for the Total Path Attribute Length (the value of Withdrawn Routes Length is 0 and the value of Total Path Attribute Length is 0).

Rekhter, et al. Standards Track [Page 20]

RFC 4271 BGP-4 January 2006

An UPDATE message can advertise, at most, one set of path attributes, but multiple destinations, provided that the destinations share these attributes. All path attributes contained in a given UPDATE message apply to all destinations carried in the NLRI field of the UPDATE message.

An UPDATE message can list multiple routes that are to be withdrawn from service. Each such route is identified by its destination (expressed as an IP prefix), which unambiguously identifies the route in the context of the BGP speaker - BGP speaker connection to which it has been previously advertised.

An UPDATE message might advertise only routes that are to be withdrawn from service, in which case the message will not include path attributes or Network Layer Reachability Information. Conversely, it may advertise only a feasible route, in which case the WITHDRAWN ROUTES field need not be present.

An UPDATE message SHOULD NOT include the same address prefix in the WITHDRAWN ROUTES and Network Layer Reachability Information fields. However, a BGP speaker MUST be able to process UPDATE messages in this form. A BGP speaker SHOULD treat an UPDATE message of this form as though the WITHDRAWN ROUTES do not contain the address prefix.

3.3 KEEPALIVE message

4.4. KEEPALIVE Message Format

BGP does not use any TCP-based, keep-alive mechanism to determine if peers are reachable. Instead, KEEPALIVE messages are exchanged between peers often enough not to cause the Hold Timer to expire. A reasonable maximum time between KEEPALIVE messages would be one third of the Hold Time interval. KEEPALIVE messages MUST NOT be sent more frequently than one per second. An implementation MAY adjust the rate at which it sends KEEPALIVE messages as a function of the Hold Time interval.

If the negotiated Hold Time interval is zero, then periodic KEEPALIVE messages MUST NOT be sent.

A KEEPALIVE message consists of only the message header and has a length of 19 octets.

3.4 NOTIFICATION message

NOTIFICATION Message Format

A NOTIFICATION message is sent when an error condition is detected. The BGP connection is closed immediately after it is sent.

Rekhter, et al. Standards Track [Page 21]

RFC 4271 BGP-4 January 2006

In addition to the fixed-size BGP header, the NOTIFICATION message contains the following fields:

0 1 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 +---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+ | Error code | Error subcode
| Data (variable) | +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
---+---+

Error Code:

This 1-octet unsigned integer indicates the type of NOTIFICATION. The following Error Codes have been defined:

Error Code Symbolic Name Reference

1 Message Header Error Section 6.1

2 OPEN Message Error Section 6.2

3 UPDATE Message Error Section 6.3

4 Hold Timer Expired Section 6.5

5 Finite State Machine Error Section 6.6

6 Cease Section 6.7

Error subcode:

This 1-octet unsigned integer provides more specific information about the nature of the reported error. Each Error Code may have one or more Error Subcodes associated with it. If no appropriate Error Subcode is defined, then a zero (Unspecific) value is used for the Error Subcode field.

Message Header Error subcodes:

1 - Connection Not Synchronized. 2 - Bad Message Length. 3 - Bad Message Type.

Rekhter, et al. Standards Track [Page 22]

RFC 4271 BGP-4 January 2006

OPEN Message Error subcodes:

1 - Unsupported Version Number. 2 - Bad Peer AS. 3 - Bad BGP Identifier. 4 - Unsupported Optional Parameter. 5 - [Deprecated - see Appendix A]. 6 - Unacceptable Hold Time.

UPDATE Message Error subcodes:

1 - Malformed Attribute List. 2 - Unrecognized Well-known Attribute. 3 - Missing Well-known Attribute. 4 - Attribute Flags Error. 5 - Attribute Length Error. 6 - Invalid ORIGIN Attribute. 7 - [Deprecated - see Appendix A]. 8 - Invalid NEXT_HOP Attribute. 9 - Optional Attribute Error. 10 - Invalid Network Field. 11 - Malformed AS_PATH.

Data:

This variable-length field is used to diagnose the reason for the NOTIFICATION. The contents of the Data field depend upon the Error Code and Error Subcode. See Section 6 for more details.

Note that the length of the Data field can be determined from the message Length field by the formula:

$$\text{Message Length} = 21 + \text{Data Length}$$

The minimum length of the NOTIFICATION message is 21 octets (including message header).

3.5 BGP Communities

Es un atributo, tag o label

[3]

LINKS

- Como funciona BGP, muy buena explicacion: <https://blog.cloudflare.com/es-es/prepends-considered-harmful/>

Capítulo 4

AS Inference

- These routing policies are constrained by the contractual commercial agreements between Administrative domains [GAO]
- Routing policies are mainly determined by the business relationships between ASes [ruan]
- Since 2 ISPs might merge into one and each administrative domain can poses several ASes, an adminis domain can operate one or several ASes. [4]
- Routing policies are mainly determined by the business relationships between ASes [ruan]

(BGP permite que cada sistema autónomo (AS) elija su propia política administrativa al seleccionar rutas y propagar información de alcance a otros. Estas políticas de enruteamiento están restringidas por los acuerdos comerciales contractuales entre dominios administrativos.)

- Las políticas de enruteamiento de BGP están determinadas principalmente por las relaciones comerciales entre ASes vecinos
- Dado que las relaciones entre ASes no son de acceso público, varios estudios han propuesto algoritmos heurísticos para inferir dichas relaciones utilizando datos de BGP disponibles públicamente.
- La mayoría de estos algoritmos se basan en la propiedad libre de valles de los caminos AS. Sin embargo, no todos los caminos AS cumplen esta propiedad, ya que algunos ASes no se ajustan a la política de exportación común. Como resultado, las relaciones inferidas entre ASes suelen ser inexactas. (e ha demostrado que si todos los ASes se

adhieren a esta política de exportación común, entonces todos los caminos AS estarán libres de valles.)

Gao fue la primera persona en plantear el problema de inferencia de las relaciones entre Sistemas Autónomos. En su estudio [4], sentó las primeras bases para tratar el problema como:

- Propuso una representación de forma de grafo para la representación de las relaciones entre los Sistemas Autónomos. Entre las que se encontraban relaciones de Customer-Provider, Peering y Sibling.
- Propuso un algoritmo heurístico para inferir las relaciones entre ASes a partir de tablas de enrutamiento BGP.

Para Gao esto definía a un Customer como un AS que paga a un proveedor para obtener conectividad al resto del Internet, Provider como aquel que transita tráfico de sus clientes, relación de peering como aquellos que se ponen de acuerdo de transitar tráfico de sus clientes entre ellos. Como el ruteo entre ASes es controlado por el protocolo de ruteo BGP, conectividad física entre dos ASes no significa reachability.

4.1.1 Gao

El algoritmo presentado por Gao Fue la primera en abordar el problema de inferencia de los tipos de relaciones entre Sistemas Autónomos. a partir de tablas de enrutamiento BGP y la heurística de los grados de cada AS.

Para validar sus resultados, Gao utilizó información interna de AT&T y datos del servicio WHOIS para verificar relaciones de tipo Siblings. En este análisis, el 99.1% de los resultados de inferencia fueron confirmados por AT&T.

Su algoritmo consistió en : 1.- **Parsear las Routing Tables**: Se parsean las tablas de enrutamiento BGP y se calcula el grado de cada AS. 2.- **Identificar el top provider**: Se identifica el top provider de cada AS path. 3.- **Clasificar las relaciones**: Se clasifican las relaciones entre los ASes en base a la jerarquía de los ASes y el top provider.

La clasificación de relaciones estaba basada en la lógica de que una vez encontrado el top provider de un AS path, los pares AS que se encontraran antes del top provider tendrían una relación de P2C, mientras que los pares que se encontraran después del top provider tendrían una relación de P2P y relación S2S si ambos pares proveían de transito para each other es decir, podía encontrarse ese par tanto antes como después del top provider.

Pero como no se puede asumir que todos los routers que pertenecen a un AS pueden estar bien configurados, Gao propuso una mejoría para evitar inferencias incorrectas la cual consistía en contar un número de Routing tables que que concluían que una relación era de un tipo y otra. Es decir, si no más de L routing tables inferían que un AS u proveía de tránsito a AS y más de L routing tables inferían que v proveía de tránsito a u entonces se ignoraba que u proveía de tránsito a v, siendo L una constante pequeña.

En caso de las relaciones de P2P Gao primero identificó aquellos ASes con quienes no podría establecer relaciones de peering las identificó bajo la idea de que en una relación de peering los ASes no differ more than R times.

4.1.2 Lu Rank

Uno de los algoritmos más recientes es el propuesto por Ruan y Susan en 2014 [5]. Hasta ese momento, la mayoría de los métodos para inferir los tipos de relaciones entre sistemas autónomos se basaban en la suposición de que todos los AS siguen una política uniforme de exportación de rutas. Según esta política, las rutas provenientes de proveedores y peers no son exportadas AS vecinos que también son proveedores o peers. Bajo esta idea, todos los AS Paths serían «*valley-free*».

Los algoritmos previos asumían que todos los AS paths eran «*valley-free*» o buscaban maximizar el número de caminos que cumplían esa propiedad. Sin embargo, se sabe que un gran número de AS paths en los updates o tablas de enrutamiento BGP no son «*valley-free*». Esto hace que al depender de dicha propiedad las relaciones inferidas sean imprecisas.

Ruan y Susan en lugar de basarse en la propiedad de «*valley-free*», propusieron un método para clasificar las relaciones observadas entre AS, sustentándose en las relaciones de tránsito entre ellas, a través de los datos de BGP. A diferencia del algoritmo de Gao [4] [TODO:], que se basa principalmente en el grado de los AS, este método utiliza el concepto de grado de tránsito de los ASes en conjunto con la identificación de los top providers de los AS Path, permitiendo identificar las relaciones entre AS. .

El algoritmo propuesto consta de tres fases principales:

1. **Preprocesamiento de los datos:** La entrada consiste en un conjunto de BGP routing table dumps obtenidos de RouteViews [6]. Estos datos se sanitizan eliminando loops, descartando ASN inválidos, eliminando ASN duplicados y excluyendo conjuntos de ASN terminales, en caso de que estén presentes en las secuencias.

2. Procesamiento de AS paths que contienen AS Tier-1:

En esta fase se definen dos contadores $\text{cnt}(X, Y)$ y $\text{cnt}(Y, X)$, los cuales indican el número de veces que X actúa como proveedor de tránsito para Y , y el número de veces que Y actúa como proveedor de tránsito para X respectivamente. Si X e Y aparecen en un AS path antes del top provider, se interpreta que Y proporciona tránsito a X ; de lo contrario, X proporciona tránsito a Y . A partir de esta información y la identificación de ASN Tier-1, las relaciones se clasifican en base a las siguientes reglas:

- Si $\text{cnt}(X, Y) > 0$ y $\text{cnt}(Y, X) = 0$, se establece una relación P2C entre X y Y .
- Si $\text{cnt}(X, Y) > 0$ y $\text{cnt}(Y, X) > 0$, y al menos uno entre X o Y es un AS Tier-1, la relación entre ellos es P2P.
- Si $\text{cnt}(X, Y) > 0$ y $\text{cnt}(Y, X) > 0$, y ninguno de los dos es un AS Tier-1, la relación es S2S.

3. **Clasificación de AS paths indeterminados:** En esta fase final, primero se determina el top provider de los AS paths donde no se encontraron AS Tier-1. Para ello, se construye un grafo dirigido en el que cada nodo tiene un atributo denominado *distance*, que indica el camino más corto en hops a un AS Tier-1. Luego, con esta información al igual que en la fase anterior se puede determinar las relaciones de los AS paths bajo las siguientes reglas:

- Si $\text{cnt}(X, Y) > 0$ y $\text{cnt}(Y, X) = 0$, entonces X y Y tienen una relación provider-to-customer (P2C).
- Si $\text{cnt}(X, Y) > 0$ y $\text{cnt}(Y, X) > 0$, entonces X y Y tienen una relación peer-to-peer (P2P).

Esta última regla puede corresponder también a una relación S2S, en caso de que en el paso anterior se hayan clasificado como S2S. Si un AS path contiene una relación P2P que no es adyacente a un top provider, entonces se reclasifica como S2S, dado que el enlace es visible a través de un upstream provider. caso contrario se mantiene como P2P.

4.1.3 BGP2Vec

En 2020, Tal Shapira y Yuval Shavitt presentaron un nuevo enfoque [7], para la inferencia de Sistemas Autónomos utilizando por primera vez técnicas de Deep Learning. Este método se realiza se basa en la creación de embeddings de estos sistemas autónomos utilizando únicamente anuncios BGP provenientes de datasets públicos. El modelo alcanzó una precisión del 95.8 % en la clasificación de relaciones entre ASes.

El pipeline que siguieron para esta técnica consistió en dos etapas: La primera, donde se entrena un modelo de Deep Learning para generar embeddings de los ASes, y la segunda, donde se entrena un modelo de Deep Learning para clasificar las relaciones entre los ASes.

El entrenamiento de BGP2Vec para la generar embeddings de Sistemas Autónomos consiste en utilizar como entrada un vector one-hot representante del ASN (Número de Sistema Autónomo) y como salida otro vector one-hot que representa los ASNs del contexto. Luego mediante el descenso de gradiente, se ajustan los pesos de la red para maximizar la probabilidad logarítmica de cualquier ASN de contexto dado el ASN de entrada.

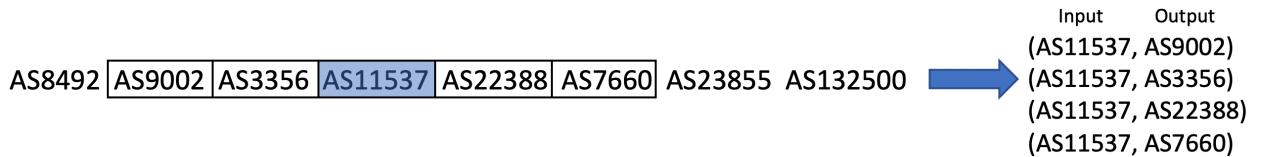


Figura 4.1: ???.

Una vez entrenado los embeddings para cada ASN, se continua con la tarea de clasificación de las relaciones entre los Sistemas Autónomos. Para esto se ocupa un Red Neuronal Artificial de dos capas.

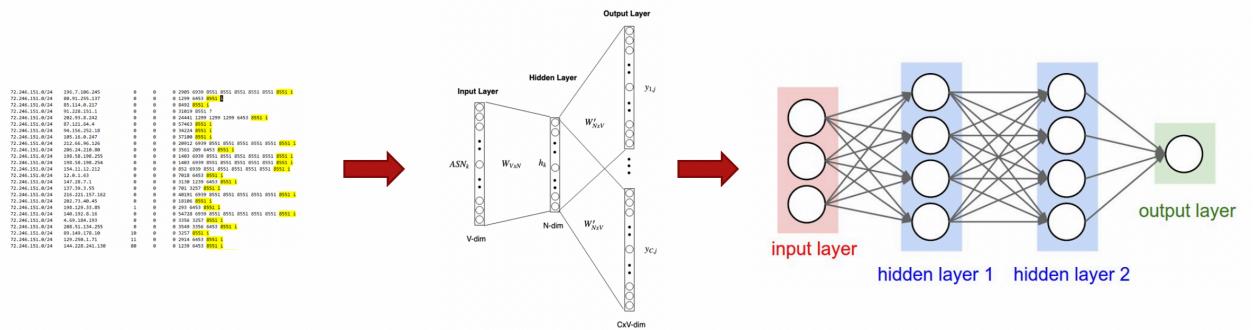


Figura 4.2: ???.

Para la etapa inicial de entrenamiento de BGP2Vec, se utilizaron anuncios extraídos de RouteViews [6], que contenían anuncios de rutas de AS (AS PATH) recolectadas por 19 colectores. Este dataset recopiló 3,600,000 rutas de AS con 62,525 vértices de AS y 113,400 enlaces no dirigidos.

Para el entrenamiento de la Red Neuronal encargada de la clasificación de las relaciones entre ASes, se utilizó el conjunto de datos de relaciones entre AS de CAIDA [8], que con-

tiene relaciones P2P y P2C/C2P. Además para algunos experimentos, se empleó el dataset ToR de BGProtect (www.BGProtect.com) [9], basado en el trabajo de Shavitt et al. [10]..

<https://github.com/talshapira/BGP2Vec/tree/main>

¿Cómo se creó el Dataset?

Para este caso se necesitan 2 datasets, el primero el dataset correspondiente a las rutas BGP, para por medio de Word2Vec characterizeembedding de los Ases y un segundo dataset para entrenar la red neuronal srtificial para la tarea de clasificacion.

1. A partir del dataset de rutas BGP proporcionadas por CAIDA [8], para el caso específico de este problema *aaaammdd.as-rel2.txt.bz2* correspondiente a la fecha dd/mm/aaaa, se generaron dos listas: una con pares de Sistemas Autonomos que comparten algún tipo de relación, y otra que etiqueta el tipo de la relación de cada par.

```
tor_dataset = []
tor_labels = []

with bz2.open(DATA_PATH + ToR_CSV, "rt") as csv_file:
    reader = csv.reader(csv_file, delimiter='|')
    for i, row in enumerate(reader):
        if row[0][0] != '#' and int(row[2]) in TOR_CSV_LABELS_DICT.values():
            # print(row)
            # Agrego arista en ambos sentidos
            tor_dataset.append(np.asarray(row[:2]))
            tor_dataset.append(np.asarray(row[1:-1]))

        # Si etiqueta -1 -> 2; si etiqueta 0 -> 0; si etiqueta 1 -> 1
        label = int(row[2])
        if label == -1:          # Si es P2C,
            label = 2
            tor_labels += [label, 1]
        elif label == 0:          # Si es P2P
            tor_labels += [label, label]
        else:                   # Si es C2P
            tor_labels += [label, 2]
```

Estas listas se gurdan en archivos np con los nombres *bgp2vec_caida_tor_dataset.npy* y *bgp2vec_caida_tor_labels.npy* respectivamente.

Las etiquetas correspondientes a las relaciones entre ASes corresponden a:

- 0: Peer-to-Peer (P2P)

- 1: Customer-to-Provider (C2P)
- 2: Provider-to-Customer (P2C)

Con un conteo de 619648, 121096 y 121096 relaciones, respectivamente.

2. Crear

3. Se entrena el modelo BGP2VEC con el dataset de rutas BGP y se guarda el modelo en el archivo *bgp2vec.word2vec*.

```
test_limit = 2000000 #cantidad paths/horaciones
mode = None# 'test' # par limitar cantidad de paths/oraciones
epochs = 1
debug = True

# Crea embeddings de ASN con BGP2VEC y lo guarda en "bgp2vec/bgp2vec.word2vec"
bgp2vec = BGP2VEC(model_path = MODELS_PATH ,oix_path=oix_path,rewrite=True,
test_limit= test_limit, mode = mode, epochs = epochs)
```

4.

Capítulo 5

Redes Neuronales

5.1 Grafos

Un grafo (Figura 5.1) es una estructura discreta formada a partir del conjunto de vértices (también conocido como nodos) y aristas las cuales son las uniones entre estas [11]. De forma más sencilla un grafo es una representación visual de una relación binaria. Un gráfo G se caracteriza mediante la pareja de conjuntos (V, E) donde V es el conjunto no vacío de vértices y E denota el conjunto de aristas, este último es, a su vez, es un conjunto de pares de nodos. Así, la definición de un grafo está dada por $G = (V, E)$. Usamos $v_i \in V$ para denotar que un nodo forma parte del grafo y $e_{ij} = (v_i, v_j) \in E$ para indicar que existe una arista entre el nodo v_i y v_j . Cada nodo v_i tiene vecinos con los cuales comparte una arista, estos se representan de la forma $N(v_i) = \{v_j \in V : (i, j) \in E\}$. El número de vértices y aristas en un grafo se representan mediante $n = |V|$ y $m = |E|$.

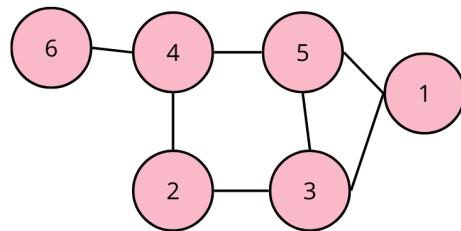


Figura 5.1: Representación de un grafo no dirigido de 6 nodos numerados.

Una forma de representar un grafo es mediante una matriz de adyacencia denotada $A \in \mathbb{R}^{n \times n}$, donde un el valor $A_{ij} = 1$ si $e_{ij} \in E$ y $A_{ij} = 0$ si $e_{ij} \notin E$. Si la matriz es simétrica, el grafo es no dirigido; de lo contrario, se trata de un grafo dirigido.

Nodos y Aristas de un grafo pueden contener atributos. De esta manera, los atributos de los nodos pueden ser representados mediante una matriz $H_n \in \mathbb{R}^{n \times d}$ donde cada fila representa un vector de características de un nodo. En el caso de los atributos de las aristas, estos pueden ser representados por la matriz de adyacencia, en la cual, en lugar de contener 1 y 0, contiene dichos atributos.

Además, los grafos pueden clasificarse en diferentes categorías que ofrecen información adicional y características distintivas. A continuación, se presentan algunas categorías comunes:

- Grafos dirigidos/no dirigidos: En un grafo dirigido, cada arista tiene una dirección específica, indicando un flujo unidireccional entre los nodos conectados. A diferencia de un grafo no dirigido, donde las aristas no tienen una orientación definida, lo que representa conexiones bidireccionales entre nodos.
- Grafos homogéneos/heterogéneos: En un grafo homogéneo, tanto nodos como aristas son del mismo tipo, en contraste de grafos heterogéneos donde los nodos y aristas pueden ser diferentes y por tanto representar cosas diferentes.
- Grafos estáticos/dinámicos: Un grafo dinámico experimenta cambios en su estructura a medida que transcurre el tiempo, a diferencia de un grafo estático, que mantiene una topología constante en función del tiempo.

5.2 Inteligencia Artificial

Inteligencia Artificial (IA) es un campo de la informática que busca simular el comportamiento de la inteligencia humana, es decir, intenta replicar y automatizar la capacidad del ser humano para tomar decisiones.

Dentro del área de la Inteligencia Artificial, nos encontramos con el Machine Learning, disciplina que a través del desarrollo de algoritmos y modelos busca que las máquinas aprendan patrones por medio de la experiencia, la cual incluye datos de entrenamiento y retroalimentación. El objetivo es entrenar una máquina para una tarea específica sin la necesidad de programar explícitamente un algoritmo.

Finalmente, dentro de Machine Learning se encuentra el campo de Deep Learning, un área que se centra en el uso de arquitecturas de Redes Neuronales profundas para aprender representaciones de datos de manera jerárquica. A diferencia de Machine Learning convencional, donde las características se extraen manualmente de los datos y se proporcionan al modelo, en Deep Learning, estas representaciones se aprenden de forma automática mientras el modelo lleva a cabo la tarea asignada. Una característica distintiva de esta disciplina es el uso de Redes Neuronales, estructuras compuestas por múltiples capas entre la entrada y la salida. Cada capa procesa la información y extrae características cada vez más abstractas a medida que se profundiza en la Red. Permitiéndole al modelo así capturar patrones y características complejas en los datos.

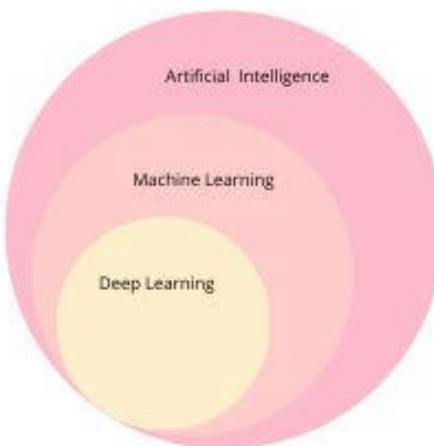


Figura 5.2: jerarquía conceptual entre Inteligencia Artificial, Machine Learning y Deep Learning.

1. Machine learning es una rama de la inteligencia artificial, que ya no depende de unas reglas y u programador, sino que la computadora puede establecer sus propias reglas y aprender por si misma
2. Aprendizaje Supervisado:

5.3 Redes Neuronales

Una Red Neuronal es un modelo computacional compuesto de neuronas (perceptrones), dispuestas en capas y conectadas entre si con el fin de aprender patrones mediante el intercambio de información ponderada por pesos. Estos pesos se ajustan en base a los datos de entrada, asignando valores en función del reconocimiento de patrones, que permiten una salida esperada.

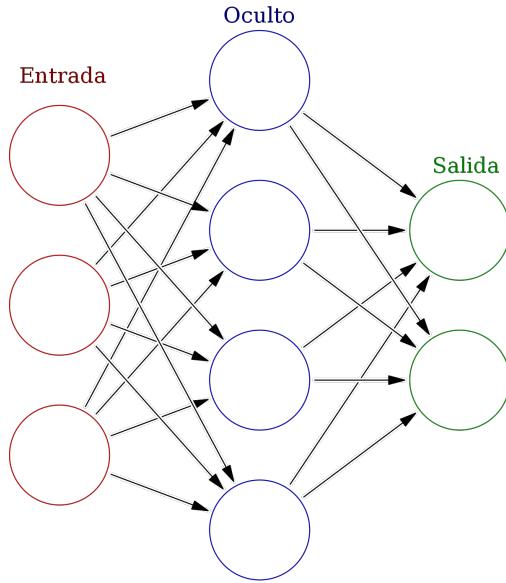


Figura 5.3: Arquitectura básica de una Red Neuronal Fully Connected de una capa.

En una Red Neuronal, cada unidad toma entradas, las pondera por separado, suma los valores y pasa esta suma a través de una función para producir una salida, la cual es compartida con otras neuronas a las cuales está conectada. El perceptrón, que funciona como una representación matemática de una unidad básica en la Red, realiza cálculos para determinar tendencias en los datos de entrada, asignándole diferentes pesos a cada valor de entrada en base a patrones entre los datos para realizar tareas específicas.

Un perceptrón está compuesto por cuatro elementos distintivos (Figura 5.4), i) los valores de entrada definidos como $x_i, x_{i+1}, \dots, x_{n-1}, x_n$ donde cada x_j corresponde a un vector de tamaño d , ii) los pesos definidos como $w_j \in \mathbb{W}^{n \times d}$ donde \mathbb{W} corresponde a la matriz de pesos los cuales son ajustados durante la etapa de entrenamiento de la Red, iii) la suma $z = \sum_{j=1}^d w_j x_j + b$ y iv) la función de activación, la cual establece un umbral de salida para evitar que los valores de salida se disparen. Esta función de activación permite incluir más capas y, por ende, mayor complejidad en las arquitecturas de redes que se construyan. Las funciones de activación tienen la capacidad de mejorar el aprendizaje de patrones en los datos[12]. Algunas de las funciones de activación comúnmente empleadas incluyen la Sigmoidal, la Tangente Hiperbólica (\tanh), la Rectified Linear Unit (ReLU) y la Leaky ReLU.

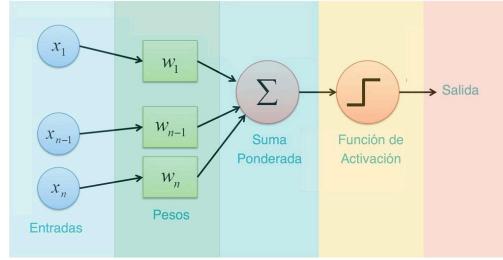


Figura 5.4: Estructura general de un perceptrón.

La técnica comúnmente utilizada para el entrenamiento de Redes Neuronales es el *back-propagation*, que tiene como objetivo ajustar los pesos de los parámetros de la Red para minimizar la función de pérdida[13]. Esta función cuantifica la diferencia entre las predicciones hechas y los valores reales. Una vez que se ha calculado la pérdida, el proceso de optimización se centra en modificar los pesos para mejorar la precisión general de la Red.

Durante el entrenamiento de las Redes Neuronales, se emplea el descenso de gradiente, un método que implica el cálculo de la derivada de la función de pérdida con respecto a los pesos de la Red. Este cálculo determina la dirección y magnitud en la que los parámetros de un modelo deben ser ajustados para minimizar la función de pérdida. Por ende, es fundamental que esta función sea continua y derivable. En problemas de regresión, se suele utilizar funciones como el Mean Squared Error y Mean Absolute Error, mientras que en problemas de clasificación, destaca la Cross-Entropy Loss.

5.3.1 Funciones de Activación

5.3.2 Función de Pérdida

5.3.3 Tipos de Redes Neuronales

5.3.4 Redes Neuronales Feed Forward (FFNN)

También conocida como *multilayer perceptrons*, esta arquitectura representa la forma más simple y fundamental de una Red Neuronal, sirviendo como la base de la mayoría de los modelos de Deep Learning. En esta arquitectura la información fluye exclusivamente hacia «adelante», sin bucles o conexiones hacia atrás.

El flujo de información comienza en la capa de entrada, donde se reciben los datos, seguida de las capas ocultas (*hidden layers* en inglés), siendo las Fully Connected las más comunes (Figura ???fully-connected), donde cada neurona está conectado a cada neurona

de la capa anterior. De esta manera, las salidas de cada perceptrón generan una salida que, al estar conectada con otros nodos, funcionan como entrada para la siguiente capa, continuando así hasta llegar a la capa de salida.

El objetivo principal de una Red Feed Forward es aproximar alguna función $f(x)$. Por ejemplo, en un problema de regresión, se busca modelar la relación $y = f(x)$.

5.3.5 Redes Neuronales Recursivas (RNN)

Las Redes Neuronales Recurrentes (RNN) son una variante de las Redes Neuronales Feed Forward, diferenciándose por su capacidad para retener y utilizar información previa, es decir, poseen «memoria».

A diferencia de las Redes Neuronales Feedforward convencionales, que asumen que los datos de entrada en cada capa son independientes entre sí, las Redes Neuronales Recurrentes (RNN) introducen conexiones entre las salidas previas y la salida actual, generando así un proceso de retroalimentación.

Esta característica en las RNN las hace particularmente eficientes para trabajar con datos secuenciales, como en aplicaciones de procesamiento del lenguaje natural incluyendo traducción, generación de texto y la predicción de series temporales.

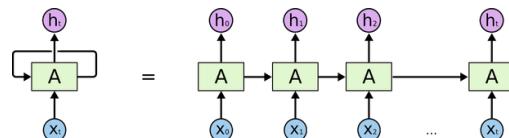


Figura 5.5: Arquitectura básica de las Redes Neuronales Recurrentes.

La imagen de arriba es una representación simple de una Red Neuronal Recurrente (Figura 5.5). En el lado izquierdo se encuentra la notación abreviada y, en el lado derecho, la notación desplegada para representar RNNs. Donde x_t es un vector que representa la entrada en el instante de tiempo t . A el estado oculto con el paso del tiempo t y actúa como la «memoria» de la Red, calculando en función del estado oculto anterior y la entrada en el paso actual.

5.3.6 Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales (Figura 5.6) son un tipo especializado de modelo de Red Neuronal diseñado especialmente para procesar información en forma de grilla [14].

Su aplicación principal se encuentra en el análisis de imágenes, en el reconocimiento de objetos, clases y categorías.

Las CNN se componen de una capa de entrada, una capa de salida y varias capas ocultas intermedias. Estas capas ocultas llevan a cabo operaciones de convolución, lo que les permite aprender características específicas de las imágenes. En el proceso de convolución, se aplican filtros, a través de matrices de pesos. Estos filtros aprenden a detectar diversas características como bordes, patrones, colores, entre otros. Así a medida que se avanza en las capas de la CNN, la red es capaz de reconocer elementos más complejos.

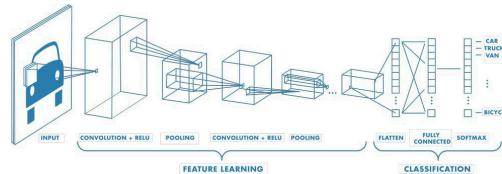


Figura 5.6: Estructura de un modelo de Convolucional con tres capas.

5.3.7 Graph Neuronla Network (GNN)

Las GNN son una arquitectura de Redes Neuronales especialmente diseñada para realizar predicciones basadas en datos representativos de grafos. A diferencia de las Redes Neuronales convencionales, las GNNs reciben datos en forma de tensores que pueden representar nodos, atributos de nodos, aristas y atributos de aristas.

Existen diferentes enfoques, dependiendo de la tarea de aprendizaje que se quiere llevar a cabo, estos son:

- Nivel de nodo: Incluye tareas como clasificación, regresión y clustering de nodos. Se realizan inferencias a partir de las conexiones con otros nodos.
- Nivel de aristas: Se abordan tareas de clasificación y predicción de aristas. Por ejemplo, determinar la existencia de una relación entre dos nodos.
- Nivel de grafo: Se encuentran tareas de clasificación, regresión y matching de grafos para las cuales el modelo debe ser capaz de aprender una representación para el grafo completo.

Las GNN tienen una serie de ventajas sobre las Redes Neuronales convencionales cuando se trabaja con datos de grafos. En contraste con los modelos tradicionales, las GNN aprovechan las relaciones entre las entidades que conforman los datos de entrada a el modelo. Estas relaciones pueden incluir aspectos como orden, jerarquía, dependencias o relaciones

de otro tipo que son comunes en grafos y se representan a través de las aristas que conectan los nodos.

En cuanto a la adaptabilidad a variaciones en el tamaño de entrada, las Redes Neuronales convencionales requieren que los datos de entrada mantengan un mismo tamaño. Para ello, recurren a técnicas como padding o broadcast, los cuales no tienen efectos significativos en el desempeño de los modelos. Las GNNs, por su parte, ofrecen flexibilidad para adaptarse a distintos tamaños de entrada[15].

Otro motivo para optar por GNNs es su capacidad para manejar el isomorfismo de los grafos, es decir dos grafos pueden lucir diferentes, pero ser estructuralmente iguales. Un modelo tradicional trataría ambos grafos como si fuesen datos diferentes, sin embargo, no lo son. Esto es comparable a lo que sucedería si se le presenta como entrada dos imágenes donde una se encuentra invertida. Es por esta razón que no se puede trabajar directamente con una matriz de adyacencia en una Red Feed Forward, ya que es sensible a estos cambios. Las GNNs utilizan técnicas que son invariantes ante permutaciones, lo que permite trabajar con el isomorfismo en grafos.

Finalmente, el último desafío radica en que la estructura de un grafo no puede ser reducida a un espacio euclidiano, y su conceptualización no puede limitarse a una distancia euclíadiana[16]. A diferencia de Redes Neuronales que trabajan, por ejemplo, con imágenes, las cuales pueden ser interpretadas como un grafo, la representación de la información se puede entender en términos de píxeles en un espacio bidimensional.

Así, la esencia detrás del uso de GNNs radica en su capacidad de entrenar un modelo que pueda procesar un grafo, sus nodos y relaciones, logrando identificar patrones relevantes en la topología para lograr de forma efectiva la tarea asignada. Por ejemplo, en el ámbito de las redes sociales, las GNNs pueden ser utilizadas para clasificar usuarios según sus interacciones, identificando así grupos afines. Otra aplicación puede ser la recomendación de contenido de interés de un usuario, basándose en sus conexiones y preferencias históricas. En el campo de la biología, es posible predecir el tipo de moléculas basándose en sus características estructurales y propiedades.

El diseño de una GNN se hace por medio de la combinación de diferentes módulos:

- Módulo de propagación: Este módulo se utiliza para propagar información entre los nodos capturando tanto la topología como los atributos de los nodos. Esto se logra combinando los datos de cada nodo con los de sus vecinos.

- Módulo de muestreo: Cuando los grafos son muy grandes, se utiliza generalmente un módulo de muestreo con el fin de seleccionar un subconjunto del grafo, aportando de este modo en la capacidad de generalización de un modelo y reducción de complejidad. Se combina generalmente con un módulo de propagación.
- Módulo de pooling: Cuando se necesita representaciones de subgrafos su utiliza este módulo para extraer información de los nodos. Se utiliza para reducir la dimensionalidad de las representaciones de nodos.

Un modelo GNN se construye generalmente combinando estos módulos. A continuación (Figura 5.7), se ilustra el pipeline de una arquitectura GNN. El modelo recibe como entrada un grafo, y en la capa GNN, se emplea un operador convolucional, un módulo de muestras y una operación skip-connection que se fusionan para propagar la información y extraer detalles de alto nivel mediante el módulo de pooling. Después de pasar por todas las capas intermedias, se obtiene una salida en forma de embeddings, a los cuales se les aplica una función de pérdida para obtener los resultados del ajuste del modelo en base a la tarea asignada.

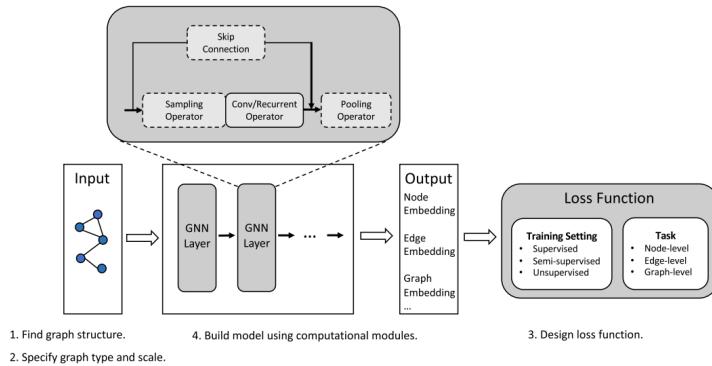


Figura 5.7: Pipeline básico de una arquitectura GNN.

5.3.7.1. Message Passing Neural Networks (MPNN)

Es la arquitectura de Red Neuronal para grafos más utilizada. Su funcionamiento radica en la idea de que cada nodo en un grafo puede intercambiar información con sus vecinos de manera que cada nodo podrá actualizar su representación en base a la información acumulada por su entorno.

La información se propaga entre nodos a través de mensajes. Cada nodo envía mensajes a sus nodos vecinos y recibe mensajes de ellos. Estos mensajes pueden contener información sobre el nodo emisor y se utilizan para actualizar la representación del nodo receptor[17].

Se emplea un mecanismo denominado *message passing*, el cual consta de tres pasos:

1. Propagación de mensajes entre nodos: Cada nodo envía un mensaje que contiene su representación actual a sus nodos vecinos.
2. Aplicación de una función de agregación: Luego de la propagación de mensajes, se aplica una función de agregación para combinar la información recibida de los nodos vecinos.

Esta función puede adoptar diversas formas como la suma o la media.

1. Actualización de la representación: La representación de cada nodo se actualiza mediante la información agregada proveniente de sus nodos vecinos, así como a partir de su representación previa.

A continuación (Figura 5.8), se presenta el comportamiento de una capa de MPNN para un nodo. El proceso inicia con el envío de un mensaje M por parte de cada nodo vecino de B . B recibe estos mensajes y los agrega mediante una operación generando una representación A . Finalmente, el nuevo estado del nodo B se calcula mediante una última función que toma el valor A y su propia representación para crear su nueva descripción U .

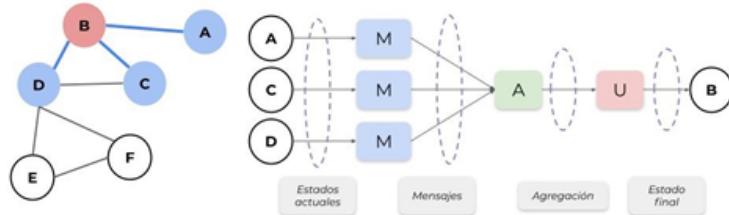


Figura 5.8: Flujo de Message passing para un nodo del grafo.

5.3.7.2. Graph Convolution Network (GCN)

Es un tipo específico de MPNN, donde se utilizan convoluciones de grafos para agregar información de los nodos adyacente de un nodo en un grafo.

La operación de convolución en el grafo produce la suma normalizada de las características de los nodos vecinos[18]. Esta normalización garantiza que la información agregada sea ponderada correctamente, es decir, evitar que un nodo con gran cantidad de vecinos tenga una representación desproporcionada y que luego tenga una influencia mayor en la representación otros nodos en las siguientes capas.

La notación de los embeddings de los nodos está dado por:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \quad (5.1)$$

Donde σ se define como la función de activación, $H^{(l)}$ la matriz de características de los nodos en la capa l , $W^{(l)}$ la matriz de aprendizaje de pesos, con dimensionalidades dada por el tamaños de atributos entrantes y de salida por capa y $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ la matriz de adyacencia normalizada.

Es así como GCN permite la creación de embeddings para los nodos de un grafo dada la matriz de adyacencia de este, lo que quiere decir que debe conocer el grafo completo para poder realizar la tarea de aprendizaje. Este es un enfoque transductivo, en contraste a otros enfoques inductivos como GraphSAGE.

5.3.7.3. Graph Attention Network (GAT)

Otra variante de MPNN son las Graph Attention Networks (GAT). A diferencia de una Red Neuronal de Convolución, GAT incorpora un mecanismo de atención que permite que cada nodo pondere de forma diferenciada, indicando la importancia de las representaciones de cada vecino para la actualización de las características de un nodo[19].

Los coeficientes se calculan por un mecanismo el cual calcula un puntaje para cada par de nodos. Luego estos puntajes se normalizan por medio de la función SoftMax (Figura 5.9).

Así tenemos:

$$z_i^{(l)} = W^{(l)} h_i^{(l)} \quad (5.2)$$

$$e_{ij}^{(l)} = \text{LeakyReLU} \left(\mathbf{a}^{(l)T} \left(z_i^{(l)} \| z_j^{(l)} \right) \right) \quad (5.3)$$

$$\alpha_{ij}^{(l)} = \frac{e_{ij}^{(l)}}{\sum_{k \in N(i)} \exp(e_{ik}^{(l)})} \quad (5.4)$$

$$h_i^{(l+1)} = \sigma \left(\sum_{j \in N(i)} \alpha_{ij}^{(l)} z_j^{(l)} \right) \quad (5.5)$$

Donde la Ecuación (5.2) corresponde a la transformación lineal del embedding de la capa anterior $h_i^{(l)}$ con $W_i^{(l)}$ una matriz de pesos entrenable.

La Ecuación (5.3) calcula un puntaje de atención entre dos vecinos. Primero concatena los embeddings z de dos nodos. Luego realiza el producto punto entre este y una matriz entrenable $a^{(l)}$ y aplica una función LeakyReLU al final.

En Ecuación (5.4) se aplica una función softmax, con el objetivo de normalizar los puntajes de atención en las aristas entrantes de cada nodo.

Finalmente, en la Ecuación (5.5), al igual que en GCN, se lleva a cabo la agregación de los nodos vecinos, pero en este caso, se escala por el puntaje de atención. Se utiliza σ como la función de activación que se aplicará a la capa.

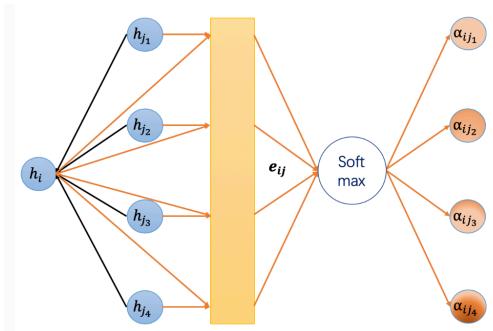


Figura 5.9: Pipeline del mecanismo de cálculo de las ponderaciones para una GAT.

5.3.7.4. GraphSAGE (SAmple and aggreGate)

Es un framework de aprendizaje inductivo el cual nos permite aprender representaciones de los nodos de un grafo. A diferencia de los enfoques anteriores los cuales son inherentemente transductivos donde se crean las representaciones de los nodos por medio de la recolección de la información de todos sus nodos vecinos, utilizando factorización de matrices, GraphSAGE «aprende» a crear las representaciones de sus nodos, es decir graphSAGE utiliza las características de nodos de su vecindario y la topología para aprender una función que genera los embeddings en base a un muestreo de nodos vecinos. Ayudado de esta forma a generalizar sobre nodos no vistos naturalmente [20]. GraphSAGE no necesita de todos sus vecinos durante el entrenamiento para crear una representación de él mismo, sino que a través de un subconjunto de estos aprenderá a crear un embedding, que representa su rol local y global en un grafo.

¿Qué significa que sea Inductivo?

Que sea inductivo significa que puede crear embeddings para nodos no vistos durante el entrenamiento. Es decir no necesita conocer todo el grafo ni todos los nodos para crear estas representaciones. Este enfoque es útil principalmente a la hora de trabajar con

grafos dinámicos, batching/sampling, etc. Representando así a nodos con un vector de baja dimensionalidad y generalizándolo para luego nodos no vistos.

El proceso de creación de embeddings para los nodos del grafo están dados por las siguientes ecuaciones:

$$h_{N(i)}^{(l+1)} = \text{aggregate}(\{h_j^l, \forall j \in N(i)\}) \quad (5.6)$$

$$h_i^{(l+1)} = \sigma(W \cdot \text{concat}(h_i^l, h_{N(i)}^{(l+1)})) \quad (5.7)$$

$$h_i^{(l+1)} = \text{norm}(h_i^{(l+1)}) \quad (5.8)$$

Donde $h_{N(i)}^{(l+1)}$ de la Ecuación (5.6) representa las características de nodos vecinos de un nodo i en la capa $l + 1$ el cual a través de una función de agregación combina estos nodos vecinos (por ejemplo promedio, suma, lstm, etc). Luego tenemos $h_i^{(l+1)}$ correspondiente a la concatenación de la representación anterior del nodo i y la de las características de nodos vecinos de la capa $l + 1$, correspondiente a lo previamente calculado. Finalmente tenemos $\text{norm}(h_i^{(l+1)})$ la cual se encarga de normalizar las características del nodo i en la capa $l + 1$.

A continuación tenemos Figura 5.10, el cual ilustra el proceso de creación de las representaciones de los nodos. Dado primero 1) por la selección de un número fijo de vecinos de un nodo, 2) Luego la agregación y concatenación de las características de estos nodos al nodo destino junto con normalización, 3) Finalmente el paso de predicción y ajuste de valores de los pesos de la red.

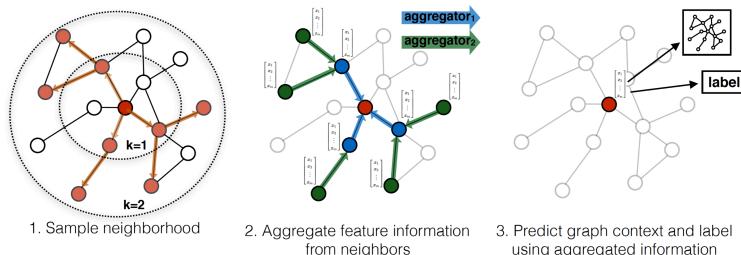


Figura 5.10: TODO.

5.3.8 Procesamiento de Datos

5.3.9 Entrenamiento

Existen 2 approach oara llevar a cabo el entrenamiento de una GNN, estos son:

- Inductive Learning: Se entrena el modelo en un subconjunto de nodos y luego se evalúa en un conjunto de nodos no vistos.
- Transductive Learning: Se entrena el modelo en todo el grafo y luego se evalúa en un subconjunto de nodos.

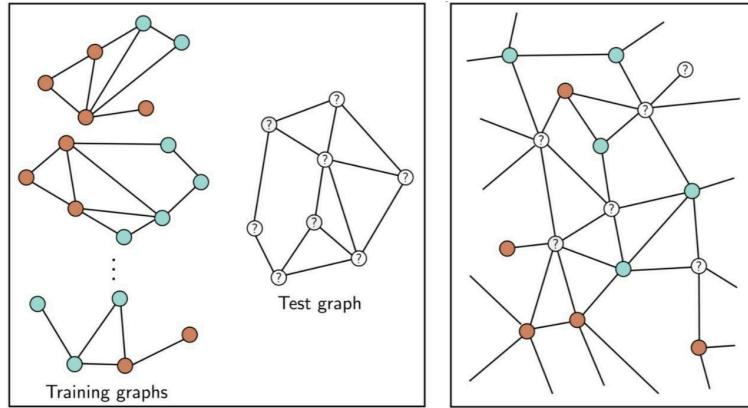


Figura 5.11: Inductive y transductive settings para entrenar y testear un modelo GNN.

En el caso de la tarea de classificación de nodos, en el enfoque inductivo, se entrena el modelo en un subconjunto de nodos y luego se evalúa en un conjunto de nodos no vistos. En cambio para el enfoque transductivo, Se tiene un solo gran grafo de donse un subconjunto de este es seleccionado para entrenar el modelo y el resto para testearlo.

5.3.9.1. Optimización

Un modelo de Deep Learning consiste en multiples capas de neuronas, las cuales se conectan entre si y organizadas en capas, estas son parametrizadas por pesos y sesgos. Estos parámetros son ajustados durante la etapa de entrenamiento de la Red, con el fin de minimizar una función de perdida, ed decir la diferencia entre la salida del doelo y los valores reales. En Figura 5.12 se muestra un esquema del proceso de entrenamiento de una Red donde tenemos nuestra Red , la cual arroja y' correspondiente a las predicciones realizadas por el modelo. Luego estos valores son pasados en conjunto con los valores reales/ esperados a la funcion de perdida, la que calcula la diferencia entre estos vallores. Con

est error se realiza el *backpropagation* calculando el gradiente de la función de perdida con respecto a los pesos de la Red.

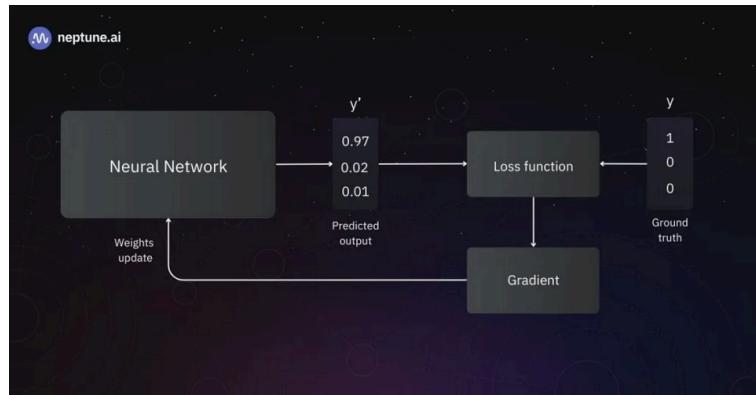


Figura 5.12: Visualización esquemática del proceso de entrenamiento en un modelo de Red Neuronal.

¿Qué es backpropagation? Es el paso en el entrenamiento de una Red donde se ajustan los parámetros de la Red en base a la función de pérdida y el algoritmo de optimización utilizado, este último se encarga de calcular el valor/cantidad que irá cambiando la Red/parámetros de la Red.

¿Qué son los optimizadores? Los optimizadores son algoritmos o métodos encargados de ajustar lo que se realiza en cada iteración de los pesos de los parámetros. Están dados por un *algoritmo de optimización*, el cual utiliza los gradientes calculados por *backpropagation* para determinar el cambio de estos pesos. Es decir, controla cómo (magnitud y dirección) de los pesos de un modelo para lograr modelar las relaciones entre los datos con el problema.

Existen diferentes algoritmos de optimización los cuales se ajustan a diferentes problemas. Estos buscan minimizar la función de pérdida, es decir llegar a un mínimo global. La decisión de esto puede estar dada por ejemplo enfocada en mejorar la precisión, reducir el tiempo de entrenamiento o gestionar los recursos computacionales. Algunos de estos son:

- Stochastic Gradient Descent (SGD)
- Mini-batch Gradient Descent
- AdaGrad (Adaptive Gradient Algorithm)
- RMSprop (Root Mean Square Propagation)
- AdaDelta
- Adam (Adaptive Moment Estimation)

Gradient Descent o Batch Gradient Descent Es el algoritmo de optimización más básico y común. comienza en un punto aleatorio y se mueve en la dirección opuesta del gradiente de la función de pérdida. Los «pasos» que van dando para acercarse a un mínimo están dados por el valor de *learning rate* en cada iteración. Se le pasa todo el dataset antes de calcular el gradiente y actualizar los pesos. Es decir se espera el paso de un forward de un epoch antes del *backpropagation* y actualización de los pesos. Hay un riesgo de overfitting ya que el modelo es expuesto de forma repetida en el mismo orden.

En Figura 5.13 muestra como el cálculo del gradiente nos permite ir avanzando en la función convexa a un mínimo, lugar donde se espera se ajuste de mejor manera los pesos de los parámetros de la Red. Se calcula el gradiente de la función de pérdida utilizando todo el conjunto de datos de entrenamiento antes de actualizar los parámetros. Esto puede requerir una gran cantidad de memoria además de ser más lento. Puede ser costoso si el dataset es muy grande (necesita realizar cálculos sobre todos los ejemplos antes de ajustar los parámetros).

Este está dado por la siguiente formulación matemática, donde se actualizan los nuevos pesos:

$$w = w - \alpha * \nabla_w J(w) \quad (5.9)$$

Donde w corresponde a los pesos de los parámetros del modelo, α el *learning rate* y $\nabla_w J(w)$ el gradiente de la función de pérdida con respecto a los pesos de la Red.

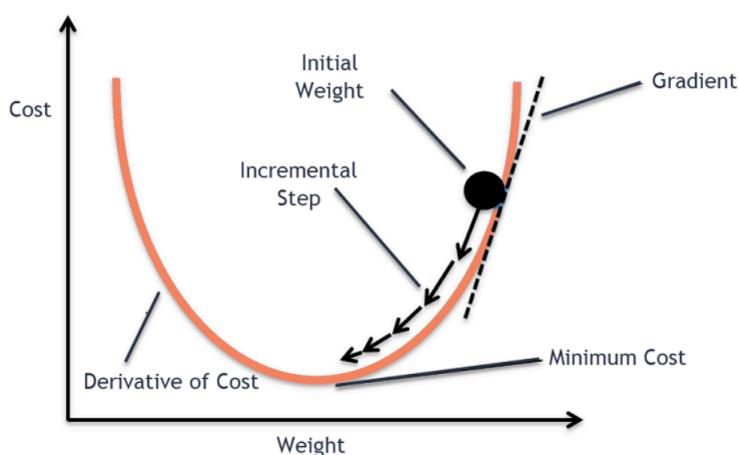


Figura 5.13: visualización optimización del algoritmo de Gradient Descent aplicada a una función convexa.

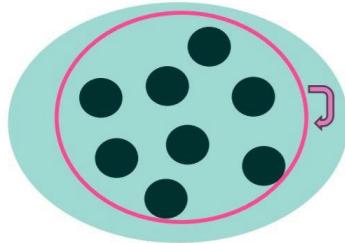


Figura 5.14: Batch Gradient Descent.

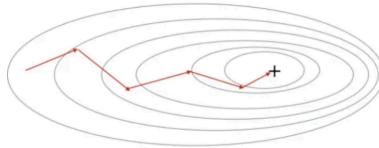


Figura 5.15: Batch Gradient Descent.

- **Stochastic Gradient Descent:**

Es una variante de la función de optimización *Gradient Descent* a diferencia de que esta actualización se realiza para cada elemento, es decir, calcula el gradiente y se realiza un update de los pesos para cada muestra del dataset.

Es por esto que los parámetros se ajustan con más frecuencia, lo que puede hacer que sea más rápido que SG y puede converger antes. Cabe mencionar que debido a la aleatoriedad de las muestras y diferencia entre estas, la trayectoria de la convergencia puede ser ruidosa, aunque ayudando a no estancarse en mínimos locales .

Su fórmula está dada por:

$$w = w - \alpha * \nabla_w J_i(w) \quad (5.10)$$

Donde w representa los parámetros del modelo, α el *learning rate* y $\nabla_w J_i(w)$ el gradiente de la función de pérdida para el i -ésimo ejemplo de entrenamiento con respecto a los pesos.

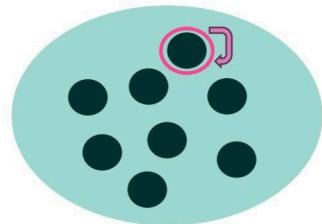


Figura 5.16: Stochastic Gradient Descent.

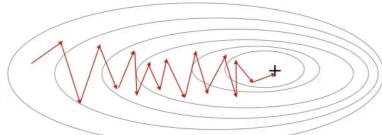


Figura 5.17: Stochastic Gradient Descent.

- **Mini Batch Gradient Descent:**

El dataset es dividido en subconjuntos llamados *mini.batches*. La actualización de los pesos de los parámetros se actualiza por mini-batch. Se introduce un nuevo hiperparámetro correspondiente al tamaño de los *mini-batches*.

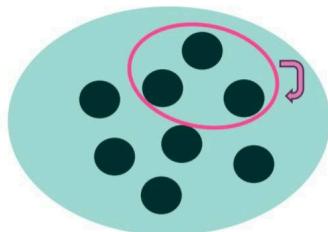


Figura 5.18: Batch Gradient Descent.

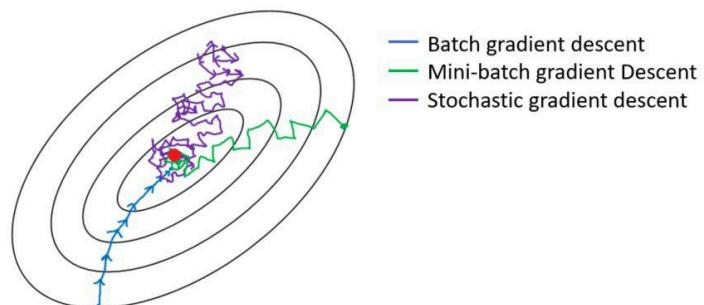


Figura 5.19: Tipos de Entrenamiento.

- **Descenso del Gradiente con Momentum:**

Introduce un término de «momentum» (inercia) en el cálculo del gradiente para evitar oscilaciones y hacer que el proceso de optimización sea más suave y eficiente. Ayuda a no quedar atrapado en mínimos locales de la función de pérdida.

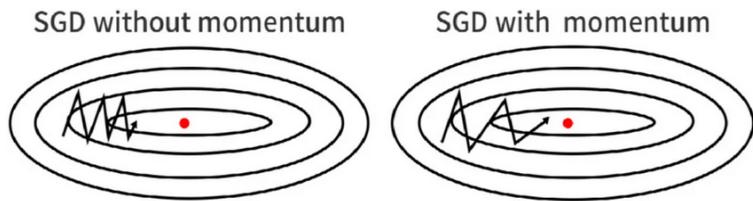


Figura 5.20: Descenso del Gradiente con Momentum:

Su fórmula matemática está dada por:

$$\nu = \nu * \eta - \alpha * \nabla_w J(w) \quad (5.11)$$

Donde η se define como el momentum, α el *learning rate* y $\nabla_w J(w)$ el gradiente de la función de pérdida con respecto a los pesos de la Red. Con esto tenemos que cada «paso» depende de los pasos anteriores ayudando a reducir ruido en la convergencia.

- **AdaGrad(Adaptive Gradient Descent):**

Ajusta dinámicamente durante el valor del *learning rate*. Asigna un *learning rate* para cada parámetro del modelo, basándose en la magnitud de los gradientes previos. A medida que se va iterando en el entrenamiento, se va acumulando el cuadrado de los gradientes de cada parámetro. Recordando así lo que ha cambiado el gradiente a lo largo del tiempo. Así parámetros con gradientes grandes tendrán una tasa de aprendizaje reducida, lo que previene actualizaciones grandes, mientras que los parámetros con gradientes pequeños se actualizan más rápidamente.

Es ineficiente para tareas que requieren un entrenamiento largo, debido a que la tasa de entrenamiento irá decayendo haciendo que luego se vuelva muy pequeña.

- **AdaDelta:** Es una mejora sobre AdaGrad, la cual toma la misma formula, pero en vez de tomar todo el historial de derivadas al cuadrado, lo toma en una ventana de tiempo (δ).

- **Adam (Adaptive Moment Estimation):** Uno de los optimizadores más utilizados, ajusta dinamicamente el *learning rate* para cada parametro. Adam toma en cuenta la evolución del *learnig rate* y la evolución de los pesos por medio de la media y varianza y los incorpora al calculo de los nuevos pesos. Combina las ideas de AdaGrad y RMSprop.

$$\text{pesos} = \text{pesos} - (\text{Momentum y varianza combinados}) \quad (5.12)$$

¿Cuál es la importancia del *Learning Rate*?

El *learning rate* es un hiperparámetro lo que quiere decir que es seleccionado manualmente, este ajusta la velocidad a la que vamos a ir dando los «pasos» en cada iteracion para acercarse al minimo de la funcion de perdida para aquellos optimizadores que no lo haedn dinamicamente durante el entrenamiento. La impirtancia d eeste radica en encontrar un valor optimo, de ocupar un *learning rate* muy pequeño puede hacer que el modelo tarde mucho en converger, ademas de quedarse en un minimo local, de lado contrario uno muy grande puede hacer que el modelo no converga o incluso diverga.

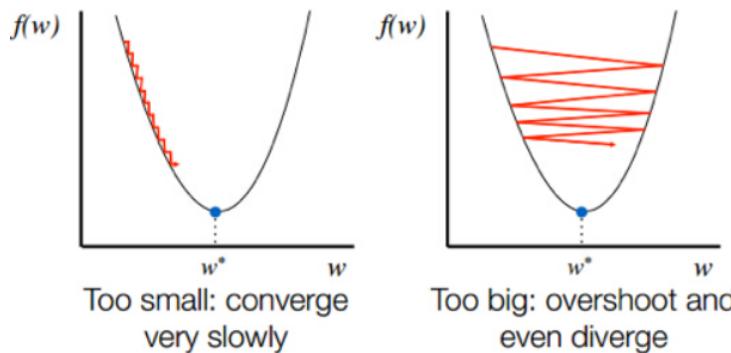


Figura 5.21: visualizaacion learning Rate grande y chico.

5.3.9.2. Sampling (Muestreo)

Sampling (muestreo en español) en Machine Learning coresponde a la tecnica utilizada para seleccionar subconjuntos de datos para entrenar o evaluar un modelo, en ve de utilizar el conjunto de datos completo. Esta tecnica se puede susar por diferentes motivos como por ejemplo dataset muy grandes y donde es computacionalmente costoso porcesar todos los datos en cada iteración del entrenamiento, otra razon es la generalización, al muestrear

diferentes subconjuntos de datos en diferentes iteraciones, el modelo tiene más probabilidades de generalizar de forma correcta y no sobreajustar los datos.

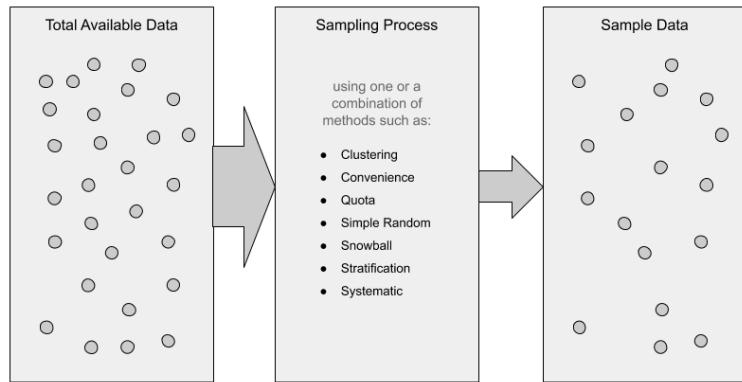


Figura 5.22: Sampling.

Existen diferentes tipos de muestreo, algunos de ellos son:

1. **Random Sampling:** Es la tecnica más simple de muestreo en Machin eLearning. Consiste en seleccionar muestras de forma aleatoria del dataset, in usar nungun patrón específico. Este metodo asume que todas las muestras del dataset son igual de importante y la misma probabilidad de ser seleccionada.
2. **Stratified Sampling:**

Esta tecnica de muestreo se usa cuando queremos tener que existan de forma equilibrada los distintos subconjuntos/estratos de datos presente en el entrenamiento, validacion y testing. Esta tecnica se usa especialmenet en tareas de clasificación para datasets desbalanceados, donde se hay mas muestras de un tipo que de otra. Para estoel dataset se divide en estratos, luego dentro de cada uno de estos subgrupos se seelcccionan muestras al azar en las mismas proporciones e que están presentes en el dataset completo.

Algunas de estas tecnicas son:

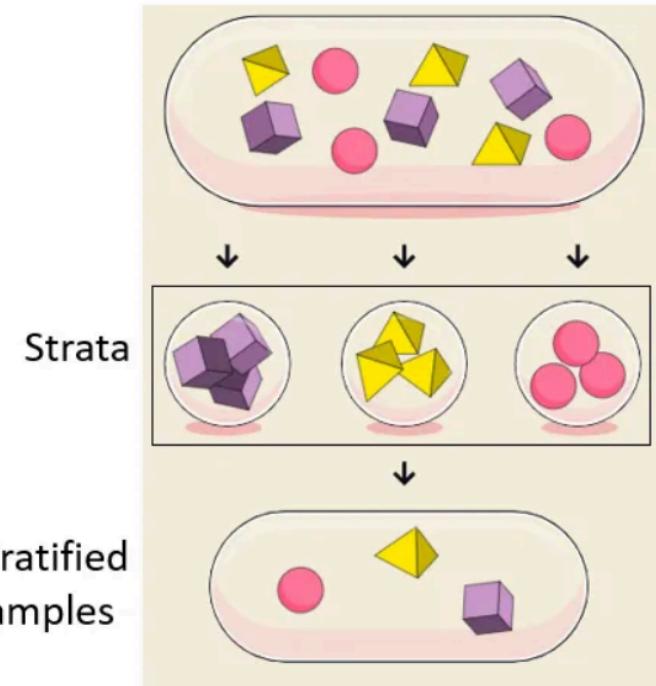


Figura 5.23: Cluster sampling.

1. **Systematic Sampling:** Es un metodo de muestreo probabilistico.

Primero se ordena el dataset de manera secuencial en base algun criterio. Luego se calculo el intervalo de muestreo $k = \frac{N}{n}$ N numero total de elementos en el conjunto y n numero de muestras aseleccionar. Se selecciona asi un punto de partida aleatorio y de ahí se seleccionan muestr as a intervalos regulares, saltando k elemntos cada vez, hasta completar las muestras deseadas.

1. **Cluster Sampling:** Tecnica usada cuando el conjunto de datos es muy grande y naturalmente está separado por grupos, donde luego se seleccionan algunso de estos conglomerados completos de manera aleatoria. A diferencia de stratified sampling donde los estratos son homogeneos dentro de un grupo, en cluster sampling estos son más heterogeneos , es decir pueden ser diferentes entre si , no se seleccionan individualmente las muestras, sino que se selecciona un grupo completo.

Es util cuando naturalmente las muestras estan divididas en conglomerados y son representativas del global. Por ejemplo: queremos obtener un sampling de poblacion, y donde cada cluster supone un grupo de personas que viven en una misma area. Se seleccionan algunos de estos clusters de manera aleatoria y se toman todas las muestras de este grupo para el estudio.



Figura 5.24: Cluster sampling.

Para el caso de Graph Neural Network existen tecnicas especificas debido a que estamos trabajando con grafos y las muestras seleccionadas dependiendo de la tarea serán nodos que aun asi estan conectados a otros nodos, es decir a pesar q se seleccionan unos nodos y otros no, estos forman parte de una estructura completa q al momento del entrenamiento se verán afectada por sus vecinos. Sampling en GNN es esencial debido a la naturaleza estructurada y muchas veces masiva de los grafos. Las tecnicas utilizadas en GNN se pueden dividir en 4 categorias principales: Random Node Sampling, Neighbor Sampling, Layer Sampling y Subgraph Sampling.

Methods used in training GCN are performed by selecting partial nodes in a graph as a sample based on the specific rule

Reduce the computation and storage cost for GCN training. Asegurando la eficiencia y escalabilidad en el proceso de training un modelo GCN

1. **Random Node Sampling:** Se selecciona de forma aleatoria un subset de nodos del grafo completo. Reduce el costo computacional en comparacion a entrenar todos los nodos de un grafo, sin embargo habrá redundancia al calcular los embeddings si es que dos nodos comparten el mismo vecino, el embedding de dicho nodo será calculado dos veces.
2. **Neighbor Sampling:** Se selecciona un numero específico de vecinos para cada nodo en cada capa de la Red. Esto permite reducir el costo computacional y la redundancia en el calculo de los embeddings, ya que es menos probable que en comparacion a la tecnica anterior haya redundancia a la hora de calcular los embeddings de un mismo vecino.

PAPERS: GraphSAGE[20], PAPERS: PinSAGE[21], [22]

1. **Layer Sampling:** Realiza un muestreo por capas de forma aleatoria entre las capas. Una desventaja de esta tecnica es que peude causar nodos aisaldos. Tiene como objetivo evitar el calculo redundante en el muestrei po rnodos. Permute un uso más eficiente de memoria.

Ejemplo de Papers FstGCN[23], Adaptative Sampling[24], LADIES[25]

1. **Subgraph Sampling:** Extrae subgrafos de manera aleatoria o divide el grafo original en subgrafos. Estos se entrenan como muestras de datos independientes. Reduce el tamaño significativamente de la data que la GNN tiene que procesar en cada iteracion.

PAPERS: GraphSAINT [26], ClusterGCN [27]

¿Qué es Batch Normalization? Tiene el fin de estabilizar el training de la GNN. Reescala cada salida de las layers ara que su promedio y varianza a traves del batch B

$$m_h = \frac{1}{|B|} \sum_{i \in B} h_i \quad (5.13)$$

$$s_h = \sqrt{\frac{1}{|B|} \sum_{i \in B} (h_i - m_h)^2} \quad (5.14)$$

Se calcula el mean y variance sobre $|B|$ embeddings en un batch B .

$$h_i \leftarrow \frac{h_i - m_h}{s_h + \epsilon} \quad (5.15)$$

$$h_i \leftarrow \gamma h_i + \delta \quad (5.16)$$

Normalizar los node embeddings usando el mean y variance calculado en el paso anterior.

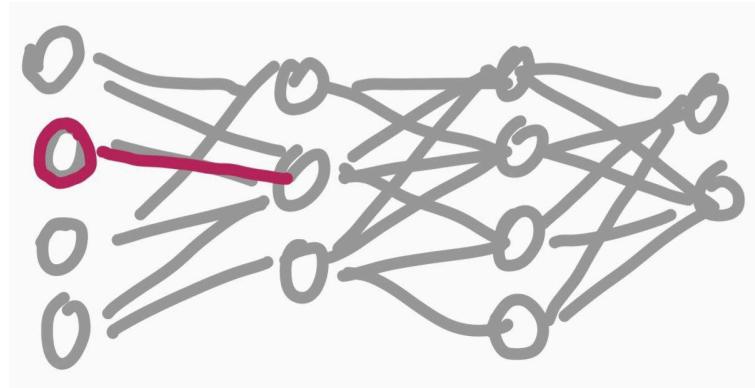


Figura 5.25: Explciacion FIXME:.

¿Qué pasa si este weigh termina siendo muy grande (encomparación a otros)? Hara que la salida d ela neurona sea muy grande, afectando asi a la siguiente capa, causando inestabilidad en el training. Nomlamente se normaliza los inputs y entonces por que no normlaizar entre capas?

- Acelera el entrenamiento (podemos ocupar lr mas grandes)
- Disminuye la imortancia de los pesos iniciales
- Regulariza el modelo (un poquito)

Entonces la idea es qu elos pesos no sean muy grandes o muy chicos y asi no se ve aafectada al estabilidad del modelo.

```

1: function BATCH NORMALIZATION(X, gamma, beta, epsilon)
2:   ▷ Calculo mean mini-batchten
3:    $\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$ 
4:   ▷ Calculo varianza del mini-batch
5:    $\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$ 
6:   ▷ Normalizar
7:    $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ 
8:   ▷ Scale and shift
9:    $y_i \leftarrow \gamma \hat{x}_i + \beta$ 
```

5.3.9.3. Regularización

Una de las metas que se tiene al momento de entrenar un modelo es evitar el overfitting y por ende que pueda generalizar los resultados de forma optima. Es decir logre la tarea correctamente en un dato nunca visto anteriormente.

Uno de los obstáculos que ocurren en gran medida causando un mal desempeño de un modelo es el sobreajuste y subajuste. Este se puede ver en Figura 5.26

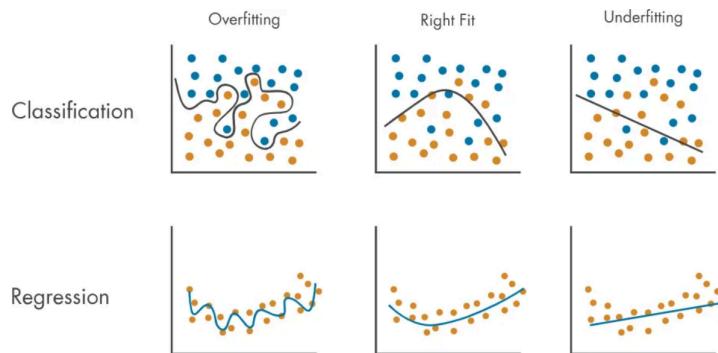


Figura 5.26: Evaluation.

Overfitting: Ocurre cuando un modelo sea ajustado demasiado a los datos de entrenamiento, de forma que se los aprende de memoria y no puede generalizar a nuevos datos. Es decir, el modelo se ajusta demasiado a los datos de entrenamiento y no puede generalizar a nuevos datos.

Underfitting: Ocurre cuando el modelo no es suficientemente complejo como para capturar la estructura subyacente de los datos. Es decir, el modelo es demasiado simple para capturar la complejidad de los datos y no puede hacer predicciones precisas.

Para poder evitar el sobreajuste y subajuste se utilizan técnicas de regularización, las cuales tienen como objetivo reducir la complejidad del modelo y evitar el sobreajuste y lograr generalizar los modelos de Deep Learning. Existen diferentes técnicas para lograr esto algunas de estas son:

- **Dropout:** Es una técnica que desactiva un número de neuronas de forma aleatoria. Para aplicar este método se asigna una probabilidad a cada neurona de ser desactivada en la fase de entrenamiento. Esto quiere decir que las conexiones que tenía esa neurona desaparecerán momentáneamente.

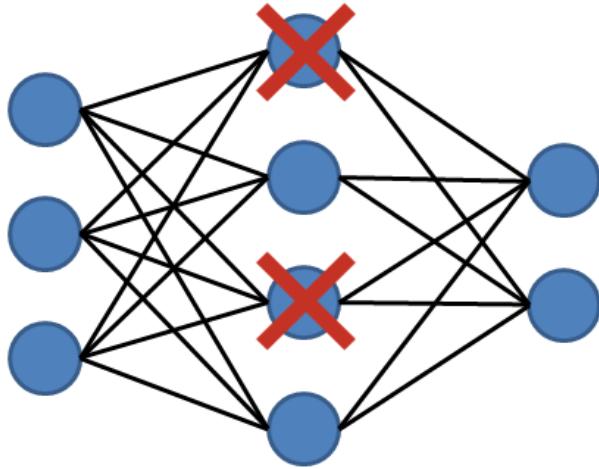


Figura 5.27: Dropout.

Para el caso de GNN En training: De forma aleatoria se seleccionan neuronas a 0 (drop out) con una probabilidad de p e una capa específica. En Testing: Se usan todas las neuronas. PAPER: [28]

- **Early Stopping:** Consiste en detener el entrenamiento antes de que el rendimiento del modelo empiece a empeorar para un conjunto de validación, evitando así un overfitting sobre los datos.
- **L1 y L2 Regularization:** Técnicas para controlar la complejidad de los modelos. Estos incluyen un término de penalización en la función de costo, penaliza los modelos con valores de coeficientes altos (y por ende complejos), esto porque modelos menos complejos son menos propensos al sobreajuste, mejorando así la generalización de este. En caso de L1 la penalización está dada por valor absoluto de los coeficientes y en L2 por el cuadrado de estos, haciendo que para el primer caso elimine características y el segundo haga una reducción más suave de la complejidad, considerando aún así todas las características.

5.3.10 Evaluación

La evaluación es el proceso de usar diferentes métricas de evaluación para comprender el rendimiento de un modelo de aprendizaje automático, así como sus fortalezas y debilidades. Esto es necesario para poder asegurar que un modelo sea confiable, generalizable y capaz de realizar predicciones correctas sobre nuevos datos no vistos.

5.3.10.1. Metricas de evaluación

Existen diferentes metricas para poder evaluar un modelo de MAchine Learning ebn base a la tarea que ester ealizará, algunas de estas son:

- Metricas de Clasificación: Accuracy, Precision, Recall,F1-score, ROC, AUC, etc.
- MEtricas de Regresión: MSE, MAE, R2, etc.
- Metricas de Ranking: MRR, DCG, NDCG, etc.
- Metricas de Estadistica: Correlación

entre otras...

Partiendo con las metricas de clasificación, primero tenemos que entender qué es la **Matriz d eConfusión**. Es una tabla que se usa para describir el rendimiento de un modelo de clasificación en el conjunto de testing d los datos. Definiendo dentro de esta los conceptos de True Positive, True Negative, False Positive y False Negative.

		true class		predicted class	total
		EFR	LFM		
predicted class	EFR	True Positives (TP)	False Positives (FP)	predicted EFR	PR = $\frac{TP}{TP+FP}$
	LFM	False Negatives (FN)	True Negatives (TN)		RE = $\frac{TP}{TP+FN}$
true		EFR	LFM	predicted LFM	
		EFR	LFM	CA = $\frac{TP+TN}{TP+TN+FP+FN}$	
				F1 = $\frac{2TP}{2TP+FP+FN}$	

Figura 5.28: Matriz de confusión.

- True Positive (TP):
- True Negative (TN):
- False Positive (FP):
- False Negative (FN):

Asi la definición de Accuracy, Precision, Recall, F1-Score y ROC para una tarea de clasificación esta dada por:

- **Accuracy:** Corresponde a la proporcion de predicciones relaizadas correctamente en comparación al numero total de predicciones realizadas. Es decir, de todas las predicciones positivas realizadas, cuantas de estas son realmente positivas.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \quad (5.17)$$

Usando la matriz de confusión:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (5.18)$$

- **Precision:** Evalua la exactitud de predicciones positivas realizadas por un modelo. Es la proporción de verdaderos positivos en proporción a las predicciones positivas realizadas por la Red. Una alta presicion indica que , cuando el modelo predice un resultado positivo, es probable que sea correcto.
- **Recall o True Positive Rate:** También conocido como Sensibilidad, mide la cpacidad del modelo para identificar todos los ejemplo positivo . Esta dado por la proporcion de verdaderos positivos sobre el total de ejemplos reales positivos. Un alto recall indica que el modelo es bueno identificando la clase positiva, es decir tiene pocas omisiones de positivos.

$$\text{True Positive Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (5.19)$$

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (5.20)$$

- **ROC :** Receiver operating characteristic. Es una representación gráfica que nos muestra en rendimiento de un modelo de clasificación en base a un umbral de desición. Este ayuda a visualizar el trade-off entre la tasa de verdaderos positivos y la tasa de falsos positivos en diferentes puntos.
- **AUC:** Area Under the Curve, es un valor que representa el area bajo la curva ROC. Mientras más cercano a 1, mejor es el modelo.



Figura 5.29: ROC Curve.

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (5.21)$$

- **F1-Score:** Esta metrica mezcla la presicion y el recall en un solo valor. De esta forma evalua el rendeminento del modelo. Esecialmente util en cpontextos de desbalance de clases

$$\text{F1 score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.22)$$

En cuanto a metricas para la Regresión, tenemos:

- **Mean Squared Error (MSE):** Es el promedio de los cuadrados de las diferencias entre los valores predichos y los valores reales. Es decir, mide la media de los errores al cuadrado.

Su formula está dada por:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (5.23)$$

- **Mean Absolute Error (MAE):** Es el promedio de las diferencias absolutas entre los valores predichos y los valores reales. Es decir, mide la media de los errores absolutos.

Su formula está dada por:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (5.24)$$

- **R2 Score:** Es una medida de cuánto se ajustan los valores predichos a los valores reales. Es decir, mide la proporción de la varianza en la variable dependiente que es predecible a partir de la variable independiente.

Su formula está dada por:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - |(y)|)^2} \quad (5.25)$$

Para GNN:

Capítulo 6

Internet Data collection

6.1 CAIDA

El Centro de Análisis de Datos de Internet Aplicados o CAIDA por sus siglas en inglés [29] es una organización de investigación cuyo principal objetivo es la recopilación y análisis de datos relacionados a la infraestructura, rendimiento y tráfico del Internet. Este funciona como un canal de difusión donde se publican y comparten tanto herramientas como investigaciones relacionadas a las redes de Internet.

En el contexto de nuestro problema de **inferencia de Relaciones entre Sistemas Autónomos**, CAIDA ofrece el dataset **CAIDA AS Relationships** [8]. Este clasifica las relaciones entre Sistemas Autónomos en dos principales tipos: Customer-to-provider (C2P) y Peer-to-peer (P2P). Estas relaciones son inferidas a partir de datos públicos de ruteoBGP ofrecidos por RouteViews[6] y RIPE RIS[30]. El dataset **CAIDA AS Relationships** está compuesto por dos subdatasets:

- **serial-1:** Contiene las relaciones inferidas utilizando un método similar al descrito en el estudio de M. Lukie et al.[31] . Esta disponible desde 1998 hasta la actualidad, con un archivo mensual. Cada archivo incluye un grafo completo derivado de instantáneas de las tablas BGP de RouteViews y RIPE RIS, tomadas cada 2 horas durante un período de 5 días.
- **serial-2:** Basado en el dataset Serial-1, este agrega enlaces inferidos mediante BGP communities, utilizando el método descrito por Vasileios et al. [32] . Está disponible desde octubre de 2015 hasta el presente, con archivos generados mensualmente.

Otro proyecto relevante de CAIDA es Archipelago (Ark) Measurement Infrastructure, una plataforma de medición distribuida a nivel global. Esta cuenta con nodos de medición ubicados en diversas zonas geográficas y topológicas para proporcionar una visión integral del estado y funcionamiento de Internet. Entre las mediciones realizadas a través de Ark se encuentran proyectos como The Spoofer Project ,Scamper, Internet Topology Discovery, entre otros.

Además, CAIDA ofrece BGPSStream [33], una librería de código abierto diseñada para manejar grandes volúmenes de datos BGP. Esta herramienta permite acceder tanto a datos BGP en tiempo real como a históricos, obtenidos de los colectores de RouteViews y RIPE NCC. Esto facilita la investigación y el análisis de eventos relacionados con el enrutamiento, como interrupciones, fugas de rutas o ataques BGP, de manera eficiente y rápida.

6.2 RouteViews Project

El proyecto RouteViews [6] de la Universidad de Oregon nació en un principio como una herramienta para los operadores de Internet, con el fin de que estos pudiesen obtener información BGP en tiempo real sobre el sistema de enrutamiento global desde las perspectivas de varios backbones y ubicaciones en Internet.

Sin embargo hoy en día, su uso se ha extendido a múltiples tareas, tales como la visualización de rutas AS, el estudio de la utilización del espacio de direcciones IPv4 y otros aspectos relacionados a la infraestructura y enrutamiento de Internet.

El proyecto cuenta con alrededor de 41 recolectores de enrutamiento (ver en ANEXO) ubicados en puntos clave alrededor del mundo, como Ámsterdam, Londres, Sidney, Singapur, São Paulo, Santiago, Nairobi, Sudáfrica, Chicago, California y otras ubicaciones, especialmente en Internet Exchange Points (IXP). Estos recolectores obtienen información de enrutamiento de más de 260 organizaciones diferentes que comparten sus datos con RouteViews, incluyendo muchos de los ISP más grandes del mundo.

Los recolectores establecen sesiones de peering BGP con diferentes sistemas autónomos, a través de estas conexiones, los recolectores reciben actualizaciones de las tablas BGP (RIS) y UPDATES del protocolo BGP, obteniendo así información sobre las rutas globales y las conexiones entre AS. En estas actualizaciones incluyen información como:

- **AS Path:** Camino de Sistemas Autónomos que un paquete debe seguir para llegar a una red destino.
- **Prefijos IP:** Rangos de direcciones IP que están siendo anunciados por los AS.

- **Next Hop:** Siguiente salto (router o red) que los paquetes deben tomar para continuar hacia su destino.

El proyecto también cuenta con una compilación de archivos en dos formatos, los obtenidos por Cisco y por el software de enrutamiento FRR. El primero recopila información en intervalos de 2 horas, comenzando a las 00:00 UTC. El segundo obtiene RIBs y actualizaciones, los RIBs corresponden a snapshots recogidas cada 2 horas, mientras que los Updates son archivos continuos que se actualizan cada 15 minutos. Estos archivos históricos están en formato MRT y están disponibles para cada recolector.

6.3 RIPE NCC RIS

RIPE (Reseaux IP Européens) Network Coordination Centre es uno de los cinco Registros Regionales de Internet (RIRs) que proporciona servicios de registro de Internet y coordinación de recursos de Internet para Europa, Oriente Medio y partes de Asia Central. RIPE NCC opera el Routing Information Service (RIS) [30], una plataforma de recopilación de datos de enrutamiento, que tiene como objetivo mejorar la comprensión y visibilidad del sistema global de enrutamiento de Internet.

Al igual que RouteViews, RIS recopila datos a través de un conjunto de más de 26 colectores (ANEXO) remos de rutas (RRCs) distfribuidos globalmente, generalmente ubicados en Puntos de Intercambio de Internet (IXPs). Voluntarios se conectan con los RRCs mediante BGP, y RIS almacena los mensajes que recibe de estos peering.

Para acceder a los datos, existen dos formas: directamente a través de archivos en formato MRT, RIS Live y RISwhois, o mediante herramientas que utilizan los datos de RIS, como RIPEstat, donde se puede encontrar información ya organizada, como la exploración del enrutamiento por país y la localización de información de contacto sobre abusos.

6.4 The peering DB

PeeringDB [34] es una base de datos de acceso libre, una iniciativa sin fines de lucro, gestionada y promovida por voluntarios, donde los usuarios comparten información sobre redes. Esta informacion facilita la interconexión global de redes en Puntos de Intercambio de Internet (IXPs), centros de datos, y además proporciona datos valiosos para investigaciones.

Fue creada con el fin de facilitar he resource was created to help support peering between networks and peering coordinators, and today, it includes a wide range of interconnection data from networks,

6.5 Hurricane Electric Internet Services

Capítulo 7

Experimentos

7.1 Benchmark

Para abordar este problema se comenzó por la creación de un Benchmark.

Para esto se probaron los siguientes métodos anteriores:

1. Gao [sacado de BGP2VEC] Paper - [4]
2. Ruan [sacado de BGP2VEC]: introduced by Ruan

and Susan Varghese [5]

7.2 Datos

- Se usó x durante lso X displaying
- ¿Cuanto se demoró?

7.3 modelos

Para bordar este problema se utiulizaron

7.4 Dataset de Evaluación

Las Relaciones entre SA son privadas, esto hace que en algoritmos de inferencia sea dificil validar, y para casos de machine learning, entrenar el modelos.

Lo primero que podríamos hacer es buscar un SA que publique sus AS relaciones. esto fue lo que hice la persona de la tesis y creé un dataset a partir de la información que provee Hurricane's BGP Toolkit.

Capítulo 8

Experimentos

Para abordar este problema se tomaron dos enfoques diferentes.

- Clasificación Binaria: Se tomaron p2c y c2p como una misma clase y las otras siendo p2p.
- Clasificación Multiclasa: Las relaciones p2c y c2p se tomaron como clases diferentes.

8.1 Dataset de Validación

Uno de los posibles caminos a ocupar fue ocupar como dataset de validación de la Tesis «BGP Data Analysis: Exploring Solutions for Autonomous Systems Relationships Inference»

Este utiliza el atributo BGP communities más la información RIB, esto con la idea de que hay algunos AS que publican sus políticas de BGP communities. Entonces se creó un JSON que contiene reglas de ciertos SA y reglas (BGP communities policy) con las que podría saber el tipo de relación que se tendría con un vecino.

Por ejemplo TODO: Agregar un ejemplo.

8.2 Experimento 1:

- GNN -> GCN

- Predictor -> DOtProduct y MLP
- Optimizador -> Adam
- Función de pérdida -> CrossEntropyLoss
- Split de edges para entrenamiento y validacion
- Stochastic Gradient Descent (SGD) con un learning rate de XXX.



Figura 8.1: Resultados.

- Problemas con el modelo:
 - Possible overfitting.

8.3 Experimento 2:

- GNN -> GraphSAGE
- Predictor -> MLP
- Optimizador -> Adam
- Función de pérdida -> CrossEntropyLoss
- Split de edges para entrenamiento y validacion
- Neighbou sampling.

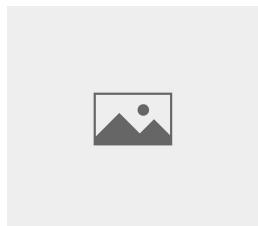


Figura 8.2: Resultados.

- ¿Por que ocupamos GraphSAGE y no GCN?

Como estamos entreando de forma transductive, es decir ocupando un mismo grafo para entrenar y validar, puede ocurrir que se este overfiteando el grafo y por eso obteniendo buenos resultados, estod ebdo a que al probar con epoch muuuuy grandes los valores de loss todo el rato eran muy similares, y lo que se espera obtener en este caso es q llegase a un punto donde la loss era similar , pero luego empezaran a diverger, que seria el punto en donde el modelo se esta empezando a aprender los datos de memoria, pero esto no

paso???. Para evitar esto se decidió ocupar otras formas de entrenamiento para mejorar que el modelo pudiera generalizar y no se estuviese validando mal (porque al final más que no es problema de que no estabamos validando con datos diferentes). Entrenamos usando ampliación.

8.4 Experimento 3:

- GNN -> GraphSAGE
- Predictor -> MLP
- Optimizador -> Adam
- Función de pérdida -> CrossEntropyLoss
- Split de edges para entrenamiento y validación
- Otro sampling



Figura 8.3: Resultados.

8.5 Experimento 4:

Agregar paquetes de flujo

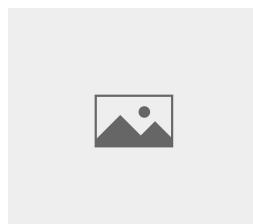


Figura 8.4: Resultados.

Destacar que en nuestro enfoque fue de Regresión, es decir la salida final de los modelos corresponde a un valor cualquiera al cual luego se calcula un umbral para clasificar en las clases correspondientes. Esto porque tener clases desbalanceadas, el modelo desearía en solo clasificar un solo tipo (la que es mayor) y aun así no se tienen resultados tan malos. Por esto no se tomó ese enfoque ya que es una de las fallas a la que queda propenso.

GCN VS GraphSAGE

- Comparamos el caso de GCN con GraphSAGE usando como Predictor el DotProduct y ocupando todas las features:

GCN:

- Con 100 epoch, un % de train de 0.6 se obtiene una accuracy de 0.81, sin embargo se puede observar overfitting mas o menos desde el epoch 70.
- Ocupando un modelo que agrega entre cada capa GNN dropout se obtiene una accuracy 0.8. No es muy diferentes.
- Drop out ayuda harto, sin embargo hace q no sea tan smooth el converger.

GraphSAGE:

- Con epoch 100, un % de experience

Podemos ver que GraphSAGE super en performance a GCN. Tambien podemos ver que para ambos modelos el overfitting se presenta en diferentes casos, esto por la naturaleza de los calculos de cada uno en la aggregación. Pues en GCN para obtener un mensaje al nodo , se necesitan todos sus vecinos haciendo que se afacil reconocer un nodo de otro ya si mas fcil de que se reconosca overfitting, en vambio en graphSAGE el mensaje al nodo en como un promedio de los vecinos, por lo que es mas dificil que se reconosca overfitting.

Se realizo la misma comparación pero en vez de DotProduct se ocupó MLP Predictor. En el caso de GCN La accuracy subió a un 0.8222(0.9221 cambiando porte capas) y GraphSAGE a un 0.92 (con el mismo numero entre capas) 0.9546 (cambiando numero de capas). Sin embargo con 100 epoch no cambiaba mucho al final. Quise ver que ocurría si ocupaba un porcentaje muy bajo de train, para ver si había overfitting, pero lo que pasó fue q igual había buenos resultados...??? ocupé el modelo sin dropout y ahí se podía ver un poco el overfitting. con una cantidad de X ejemplos de edge para train.

[Cachar porque no me esta encajando el numero de true y falses en el train msakk al hacer split del dataset]

como sea Es mejor con MLP que con DotProduct, por lo que se ocupará MLP en los siguientes experimentos.

OCUPAR TODAS LAS FEATURES ES NECESARIO?

Luego para ver que tan esenciales en la tarea era los features recolectados se probó con graphSAGE y MLP como predictor (con hasta el momento los mejores resultados que se han obtenido), con 100 epoch y un % de train de 0,6 .

Se partio ocupando únicamente como atributos de los nodos su grado in y grado out por nodo. Obteniendo una Accuracy de 0.8724 con DotProduct y 0.9290 con MLPpredictor. Con esto vemos que si bien se dan buenos resultados agregar las features Mejora la accuracy. ¿Pero son de ayuda las que le estamos pasando? Porque si bien ya tenemos que son una ayuda luego de hacer una exploración de esta (En anexo mas info) podemos notar que para muchos features para la mayoría de los nodos no se tiene información y por ende no serían muy relevantes?.

PAra esto se decidió incluir únicamente aquellas features cuya información para todos los nodos estuviera sobre 80%. Es decir existe info de la feature para el 80 % de los nodos. Esto consistieron en :

- AS_rank_numberAsns
- AS_rank_customer
- AS_rank_peer
- peeringDB_ix_count
- peeringDB_fac_count
- cti_top

Con estos usando GraphSAGE y MLP como predictor se obtuvo una accuracy de 0.9452 lo que se ve que tener todas las otras valores lo mejora pero no tanto, es decir no son tan relevantes para dicha tarea.

otro caso fue elegir únicamente[COMPLETAR]...

IDEAS FALLIDAS

- Dentro de otras ideas que se intentaron pero no funcionaron fue crear grafos a partir de los recolectores RRC, de esta forma tener diferentes grafos a partir de los cuales algunos se elijan para training y otro diferente para testeo. Sin embargo falle, no se si retomar o no.
 - otra idea fue que RIPEstat tiene una API de la cual se puede obtener información, sin embargo se demora demasiado para la cantidad de nodos que tienen por grafos.
-

Continuando con los experimentos una de los problemas que mas miedo tenia era que se entrenara y se estuviera overfitteando el grafo, porque no tenemso mas grafo que el de esa fechaa, pues esos datos (los atributos corresponden a sacados por otro paper) . Es decir estabamos tomando un enfoque inductivo. Es por esto que una podible solución que se nos ocurrio fue proabrir diferentes tecnicas de samplingm_i, area que aun sigue en investigación e inovacion en el area de GNN.

Para esto se partio con random neighbour sampling[explicado en XXX marco teorico] y se obtuvo una accuracy de 0.9414 , lo que no mejoro mucho los resultados obtenidos anteriormente. ya desde el segundo epoch hay overfitting.

se decidió probar con otro tipo dxxxx meodo el cual corrspondiente a ClusterGCN [explciado marco teorico seccion XXX] obteniendo una accuracy 0.9696 !!!!!!!!

Luego para comparar GNN con otros metods para resolver dicha tarea, se probó resolver la atrae de clasificación con PAgeRank, DeepWalk y BGP2Vec. Obteniendo Resultados XX, 0.9270 (entrenando) y XX respectivamente.

La idea es ver que tan bien se comporta el modelo en comparación con otros metodos que se han ocupado para resolver la misma tarea.

Caso	Accu-racy
GCN + DotProductPredictor	0.81
GraphSAGE + DotProductPredictor	0.9
GCN + MLPPredictor	0.9221
GraphSAGE + MLPPredictor	0.9546
GraphSAGE + MLPPredictor + in_degree y out_degree	0.9290
GraphSAGE + MLPPredictor + features sobre 80%	0.9452
GraphSAGE + MLPPredictor + Random neighbour sampling	0.9414
GraphSAGE + MLPPredictor + ClusterGCN	0.9696

Caso	Accu-racy
PageRank	0.8746
DeepWalk	0.9270
BGP2Vec	X
Crear de 0	0.9068

Con GCN el overfitting es más facil que con GraphSAGE, es de esperarxe por la formula de cada una. GraphSAGE para que se vea overfitting % de train debe ser más bajo a diferencia de GCN que puede ser alto .

Capítulo 9

Resultados y Análisis

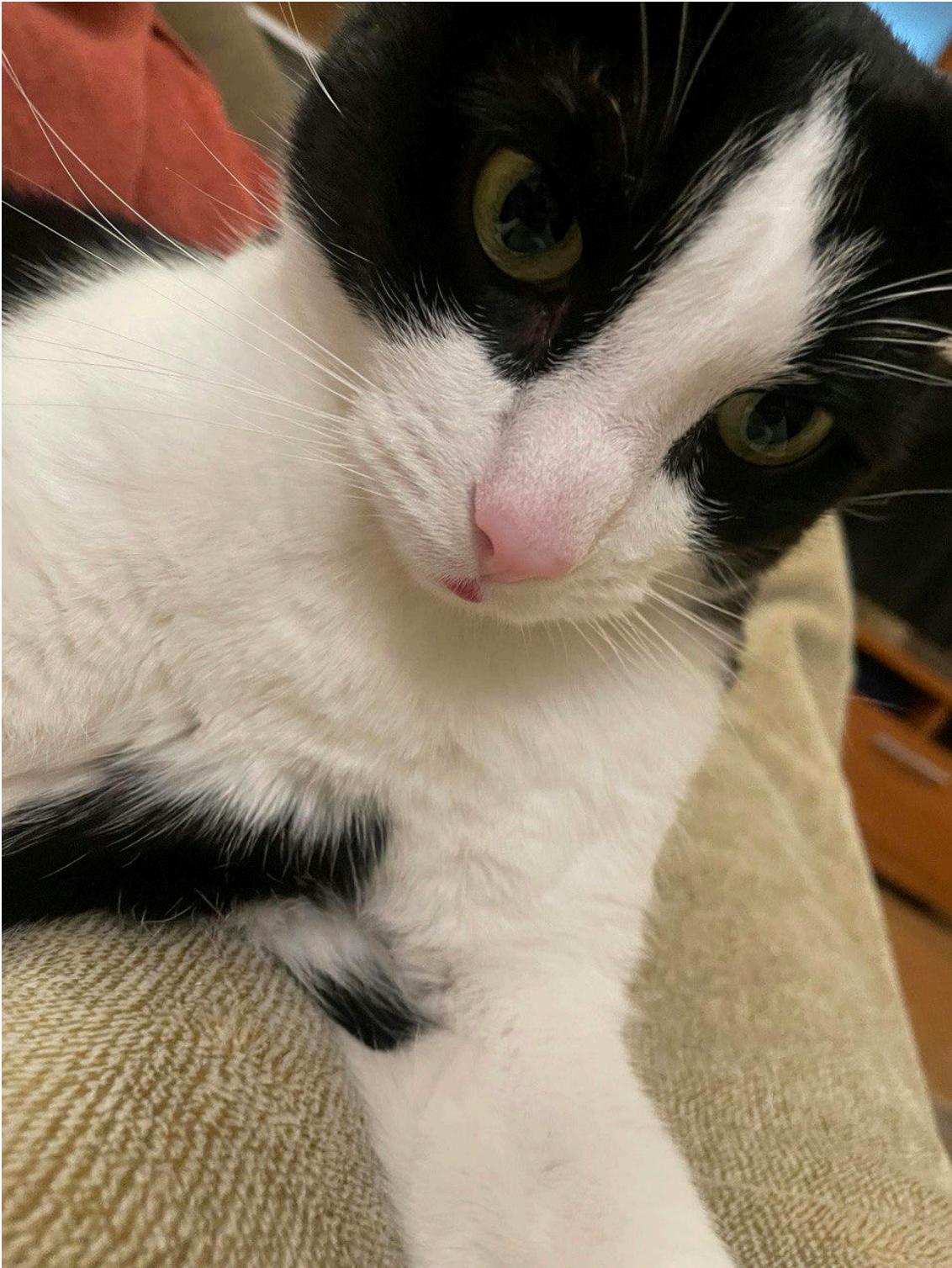


Figura 9.1: El gatito más bello del mundo.

Bibliografía

- [1] Y. S. Tal Shapira, «Unveiling the Type of Relationship Between Autonomous Systems Using Deep Learning», *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020.
- [2] E. T. L. E. S. Hares Ed. Y. Rekhter, «A Border Gateway Protocol 4 (BGP-4).», *RFC 4271*, 2006.
- [3] «BGP Communities Attribute», *RFC 1997*.
- [4] L. Gao, «On inferring autonomous system relationships in the internet», *IEEE/ACM Transactions on Networking*, 2001.
- [5] L. Ruan y J. Susan Varghese, «Computing observed autonomous system relationships in the internet». 2014.
- [6] «University of Oregon RouteViews Project, <https://www.routeviews.org/routeviews/>».
- [7] T. S. y Yuval Shavitt, «Unveiling the Type of Relationship Between Autonomous Systems Using Deep Learning». 2020.
- [8] «CAIDA. 2022. AS-relationships dataset». [En línea]. Disponible en: <https://publicdata.caida.org/>
- [9] «BGProtect, www.BGProtect.com».
- [10] U. Weinsberg Y. Shavitt y E. Shir, «Near-deterministic inference of AS relationships». 2009.
- [11] S. H., Z., Y. Z. L. L. W. C. L. M. S. Jie Zhou Ganqu Cui, «Graph neural networks: A review of methods and applications», 2018.
- [12] W. I. A. G. S. M. Nwankpa C., «Activation Functions: Comparison of trends in Practice and Research for Deep Learning. », 2018.
- [13] G. E. H. R. J. W. Rumelhart D. E., «Learning representations by back-propagating errors», 1986.
- [14] A. C. I. Goodfellow Y. Bengio, *Deep Learning*. MIT Press, 2016.
- [15] J. A.-I. R. G.-B. T. H. A. A.-G. R. P. A. David Duvenaud Dougal Maclaurin, «Convolutional Networks on Graphs for Learning Molecular Fingerprints», 2015.

- [16] Y. L. A. S. P. V. Michael M. Bronstein Joan Bruna, «Geometric deep learning: going beyond Euclidean data», *IEEE Signal Processing Magazine*, 2017.
- [17] P. F. R. O. V. G. E. D. Justin Gilmer Samuel S. Schoenholz, «Neural Message Passing for Quantum Chemistry», 2017.
- [18] M. W. Thomas N. Kipf, «SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS», 2017.
- [19] A. C. A. R. P. L. Y. B. Petar Velickovic Guillem Cucurull, «GRAPH ATTENTION NETWORKS», *International Conference on Learning Representations*, 2018.
- [20] R. Y. William L. Hamilton y J. Leskovec, «Inductive Representation Learning on Large Graphs», 2018.
- [21] K. C. P. E. W. L. H. Rex Ying Ruining He y J. Leskovec, «Graph Convolutional Neural Networks for Web-Scale Recommender Systems», 2018.
- [22] J. Z. Jianfei Chen y L. Song, «Stochastic Training of Graph Convolutional Networks with Variance Reduction», 2017.
- [23] T. M. Jie Chen y C. Xiao, «FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling», 2018.
- [24] Y. R. Wenbing Huang Tong Zhang y J. Huang, «Adaptive Sampling Towards Fast Graph Representation Learning», 2018.
- [25] Y. W. S. J. Y. S. Difan Zou Ziniu Hu y Q. Gu, «Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks», 2019.
- [26] A. S. R. K. Hanqing Zeng Hongkuan Zhou y V. Prasanna, «GraphSAINT: Graph Sampling Based Inductive Learning Method», 2019.
- [27] S. S. Y. L. S. B. Wei-Lin Chiang Xuanqing Liu y C.-J. Hsieh, «Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks», 2019.
- [28] L. F. Pál András Papp Karolis Martinkus y R. Wattenhofer, «DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks», 2021.
- [29] «CAIDA». [En línea]. Disponible en: <https://www.caida.org/>
- [30] «RIPE NCC Routing Information Service (RIS)», [https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/».](https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/)
- [31] k. c. A. D. M. Luckie B. Huffaker y V. Giotsas, «AS Relationships, Customer Cones, and Validation», in *Internet Measurement Conference (IMC)*, 2013.
- [32] M. L. Vasileios Giotsas Shi Zhou y kc claffy, «Inferring Multilateral Peering».

- [33] «BGPStream ,<https://bgpstream.caida.org/>».
- [34] «PeerinDB. 2022. The Interconnection database». [En línea]. Disponible en: <https://www.peeringdb.com/>

Anexo A

Vocabulario

- RIR: Regional Internet Registry.
- AS: Autonomous System
- BGP: Border Gateway Protocol
- RRC: Route Collectors
- IXP: Internet Exchange Point
- Internet Carrier: Empresas que operan infraestructuras de Red de gran escala. Ofrecen servicios de intercambio de datos.

Forman parte del núcleo de Internet. Ejemplos de carrier son: AT&T.

- ISP:
- CDN: Content Delivery Network
- Internet Service Provider (ISP): Proveedor de servicios de Internet. Empresa que ofrece servicios de acceso a Internet a usuarios finales. Se encuentran en la parte más periferica de la red de Internet

Anexo B

titulo anexo 1

Collectors RIPE NCC:

Nombre	Ubicación	Typo	Sponsors
RRC00	Amsterdam, NL	multihop	RIPE NCC
RRC01	London, GB	IXP	LINX, LONAP
RRC03	Amsterdam, NL	IXP	AMS-IX, NL-IX

Nombre	Ubicación	Typo	Sponsors
RRC04	Geneva, CH	IXP	CIXP
RRC05	Vienna, AT	IXP	VIX
RRC06	Otemachi, JP	IXP	DIX-IE, JPIX
RRC07	Stockholm, SE	IXP	Netnod
RRC10	Milan, IT	IXP	MIX
RRC11	New York, NY, US	IXP	NYIIX
RRC12	Frankfurt, DE	IXP	DE-CIX
RRC13	Moscow, RU	IXP	MSK-IX
RRC14	Palo Alto, CA, US	IXP	PAIX
RRC15	Sao Paolo, BR	IXP	PTTMetro-SP
RRC16	Miami, FL, US	IXP	Equinix Miami
RRC18	Barcelona, ES	IXP	CATNIX
RRC19	Johannesburg, ZA	IXP	NAP Africa JB
RRC20	Zurich, CH	IXP	SwissIX
RRC21	Paris, FR	IXP	France-IX Paris and France-IX Marseille
RRC22	Bucharest, RO	IXP	InterLAN
RRC23	Singapore, SG	IXP	Equinix Singapore
RRC24	Montevideo, UY	multihop	LACNIC region
RRC25	Amsterdam, NL	multihop	RIPE NCC
RRC26	Dubai, AE	IXP	UAE-IX, Datamena

Collectors de RRouteVIews:

- Collectors:

Host	Ubicación
amsix.ams.routeviews.org	AMS-IX Amsterdam, Netherlands
cix.atl.routeviews.org	CIX-ATL Atlanta, Georgia
decix.jhb.routeviews.org	DE-CIX KUL, Johor Bahru, Malaysia
iraq-ixp.bgw.routeviews.org	IRAQ-IXP Baghdad, Iraq
pacwave.lax.routeviews.org	Pacific Wave, Los Angeles, California
pit.scl.routeviews.org	PIT Chile Santiago, Santiago, Chile
pitmx.qro.routeviews.org	PIT Chile MX, Querétaro, Mexico
route-views.routeviews.org	Cisco IPv4 U of Oregon, Eugene Oregon
route-views.amsix.routeviews.org	AMS-IX AM6, Amsterdam, Netherlands
route-views.bdix.routeviews.org	BDIX, Dhaka, Bangladesh
route-views.bknix.routeviews.org	BKNIX, Bangkok, Thailand
route-views.chicago.routeviews.org	Equinix CH1, Chicago, Illinois
route-views.chile.routeviews.org	NIC.cl Santiago, Chile
route-views.eqix.routeviews.org	Equinix DC, Ashburn, Virginia
route-views.flix.routeviews.org	FL-IX, Miami, Florida
route-views.fortaleza.routeviews.org	IX.br (PTT.br), Fortaleza, Brazil
route-views.gixa.routeviews.org	GIXA, Ghana, Africa
route-views.gorex.routeviews.org	IGOREX, Guam, US Territories
route-views.jinx.routeviews.org	JINX, Johannesburg, South Africa

Host	Ubicación
route-views.kixp.routeviews.org	KIXP, Nairobi, Kenya
route-views.linx.routeviews.org	LINX, London, United Kingdom
route-views.mwix.routeviews.org	FD-IX, Indianapolis, Indiana
route-views.napafrica.routeviews.org	NAPAfrica, Johannesburg, South Africa
route-views.nwax.routeviews.org	NWAX, Portland, Oregon
route-views.ny.routeviews.org	DE-CIX NYC, New York, USA
route-views.paix.routeviews.org	PAIX, Palo Alto, California
route-views.perth.routeviews.org	West Australian Internet Exchange, Perth, Australia
route-views.peru.routeviews.org	Peru IX, Lima, Peru
route-views.phoix.routeviews.org	University of the Philippines, Diliman, Quezon City, Philippines
route-views.rio.routeviews.org	IX.br (PTT.br), Rio de Janeiro, Brazil
route-views.saopaulo.routeviews.org	SAOPAULO (PTT Metro, NIC.br), Sao Paulo, Brazil
route-views2.saopaulo.routeviews.org	SAOPAULO (PTT Metro, NIC.br), Sao Paulo, Brazil
route-views.sfmix.routeviews.org	San Francisco Metro IX, San Francisco, California
route-views.siex.routeviews.org	Sothern Italy Exchange (SIEX), Rome, Italy
route-views.sg.routeviews.org	Equinix SG1, Singapore, Singapore
route-views.soxrs.routeviews.org	Serbia Open Exchange, Belgrade, Serbia
route-views.sydney.routeviews.org	Equinix SYD1, Sydney, Australia

Host	Ubicación
route-views.telxatl.routeviews.org	TELXATL, Atlanta, Georgia
route-views.uaeix.routeviews.org	UAE-IX, Dubai, United Arab Emirates
route-views.wide.routeviews.org	DIXIE (NSPIXP), Tokyo, Japan

Anexo C

Cosas Extras

- ¿Cómo se ve el sobreajuste?
- Heat map para analisar de la importancia de los atributos
- ¿Por qué se obtienen mejores resultados con GraphSAGE que con CGN?
- ¿Por que no se ocupó un enfoque transductivo y únicamente uno inductivo Learning?
- Transductive
- pros y contra de cada métrica de evaluación.