UNIVERSIDAD NACIONAL DE LA PATAGONIA SAN JUAN BOSCO
Facultad de Ingeniería
Departamento de Informática.

Posicionamiento de Vértebras mediante Landmarks y Redes Bayesianas

Tesina presentada para optar al título de Licenciado en Informática.

Cintas Celia

Tutores:

Bianchi, Gloria.
Delrieux, Claudio.

Trelew, Chubut. Abril 2012.

UNIVERSIDAD NACIONAL DE LA PATAGONIA SAN JUAN BOSCO
Facultad de Ingeniería
Departamento de Informática

Posicionamiento de Vértebras mediante *Landmarks* y Redes Bayesianas

Trabajo Final de Carrera presentado para acceder al título de Licenciado en Informática.

Trelew, Chubut. Mayo 2012.

Resumen

Este trabajo pone el foco en una mejora del método tradicional, de observación de la vértebra a clasificar por parte del paleontólogo, quien mediante sus conocimientos empíricos determina una posición aproximada de la misma en la espina dorsal. El objetivo es automatizar estos pasos vía aplicaciones de la visión artificial, tales como el reconocimiento de objetos y patrones, y propone la aplicación de Morfometría Geométrica, Redes Bayesianas, y Redes Neuronales en particular, como ente clasificador, para determinar el posicionamiento de vértebras en la espina dorsal de los Saurópodos.

Como resultado se ha logrado *PyBones*, un sistema que aporta una base para actuales y futuros trabajos en la clasificación de fósiles, para lo cual implementa una base de conocimiento mediante parámetros de las imágenes obtenidas, sin necesidad de almacenar las mismas, y un agente clasificador. Cambiando los datos (*landmarks*, variaciones de los mismos y reglas) se da lugar a la clasificación de otros tipos de fósiles, como por ejemplo los dientes.

Palabras Clave: Inteligencia Artificial, Procesamiento de Imágenes, Fósiles, Redes Bayesianas, Redes Neuronales, Landmarks, Reconocimiento de Objetos, Visión Artificial.

1. Introducción

La distribución de las vértebras en una columna no es fácilmente discretizable para el ojo humano, su disposición es más bien continua, con leves variaciones. Estas variaciones deben ser determinadas por el experto en el tema y se ven reflejadas en una imagen mediante distintas distancias entre puntos específicos, un ejemplo de esto se observa en la Figura 1.

En Morfometría un punto de referencia, *landmark*, es un punto en la forma de un objeto en el que las correspondencias entre objetos y dentro de las poblaciones se preservan. Los *landmarks* pueden definirse manualmente o en forma automática, por medio de un programa computacional.

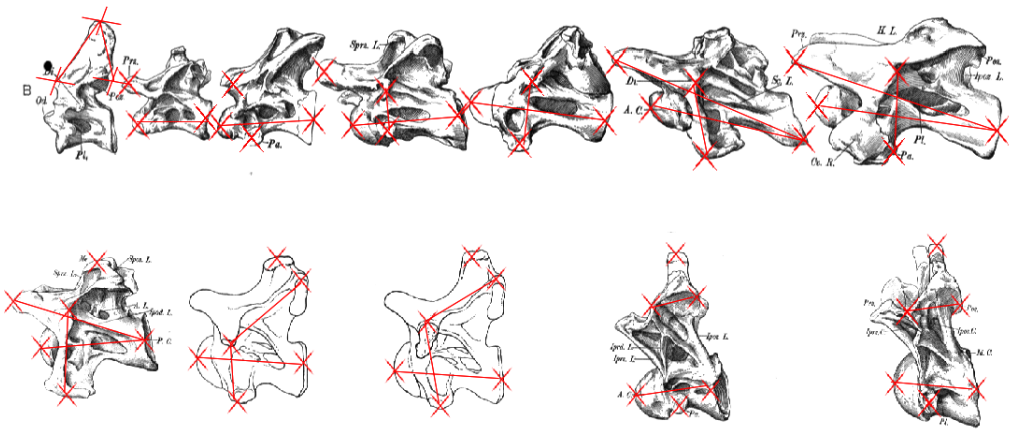

Figura 1: *Landmarks* y sus distancias observadas en vértebras cervicales.

Este trabajo pone el foco en una mejora del método tradicional, de observación de la vértebra a clasificar por parte del paleontólogo, quien mediante sus conocimientos empíricos determina una posición aproximada de la misma en la espina dorsal.

El objetivo es automatizar estos pasos vía aplicaciones de la visión artificial, tales como el reconocimiento de objetos y patrones, como se puede observar en trabajos previos realizados por [Fergus 2007] [Steger 2001] y [Szeliski 2010], entre otros. Particularmente se propone la aplicación de Redes Bayesianas [Villanueva 2007] [Juarez 2008] como ente clasificador [Felgaer 2005] [Larrain 2002], para determinar el posicionamiento de vértebras en la espina dorsal de los Saurópodos, trabajando con muestras de *Camarasaurus Supremus* y *Brachiosaurus Brancai*.

La precisión de inferencia en una Red Bayesiana, como también en la realidad, está sesgada por la cantidad de datos significativos con que se cuenta. En este punto entra en juego el aprendizaje en Redes Bayesianas, ya que a partir de bases de datos incompletas es posible inferir de alguna manera los datos ausentes para completar la base [Felgaer 2005] [Ramoni 1997].

Es así que el trabajo se centra en el aprendizaje con clasificación supervisada, desarrollado en [Forcada 2003], a partir del cual es posible realizar matching [Mery 2002] de nuevas imágenes con las preexistentes y asumir medidas de similitud para incrementar la base de conocimiento con nuevos valores.

Los datos de la base de conocimiento son proporcionados por *landmarks* tomados de imágenes que han sido previamente procesadas, y el clasificador tiene como herramientas ciertas trasformaciones que pueden aplicarse a los datos existentes, para determinar en base a las probabilidades otorgadas a cada nodo de la Red Bayesiana si el dato entrante pertenece o no a cierta ubicación en una espina dorsal.

Objetivos

Este trabajo se centra en la clasificación de vértebras, en particular de Saurópodos, pero cambiando los datos (*landmarks*, variaciones de los mismos y reglas) se da lugar a la clasificación de otros tipos de fósiles, como por ejemplo los dientes. Como objetivos de este trabajo se plantearon:

1. Desarrollar una base para actuales y futuros trabajos en la clasificación de fósiles. Para lograr este objetivo se previó:
2. Diseñar e implementar una base de conocimiento mediante los parámetros de las imágenes obtenidas, sin necesidad de almacenar las mismas.
3. Desarrollar un agente clasificador.

Contexto Actual

En el ámbito de desarrollo de software paleontológico las aplicaciones suelen orientarse a:

1. Cálculo del tiempo de crecimiento de un espécimen.
2. Implementación de bases de datos para el almacenamiento ordenado por población de individuos.
3. Manipulación de imágenes para la obtención (manual) de contornos, landmarks y su posterior análisis. (Shape[I], MorphoJ[II], PhotoModeler[III]).
4. Reconstrucción 3D.
5. Simulación de posibles capacidades motrices de un espécimen.
6. Mediciones de masa corporal, longitudes, etc.

La mayoría de las aplicaciones son de carácter privativo y costoso para los grupos de investigación. Además, el usuario debe ingresar manualmente una numerosa cantidad de datos en forma iterativa, aumentando de esta manera la probabilidad de cargar erróneamente ciertos datos.

En el caso de la mayoría de las aplicaciones mencionadas se requiere una persona que tome las decisiones al final del proceso, pero se podría implementar cierta inteligencia de clasificación en el software para agilizar el trabajo del individuo, reduciendo así la necesidad de su presencia para realizar tareas repetitivas o determinísticas bajo un conjunto cerrado de decisiones.

Otro aspecto destacable es que existen métodos y tecnologías para obtener representaciones digitales de un objeto físico, como es el caso del tomógrafo, escáner, láser, ultrasonido, los que están disponibles con un costo medianamente accesible, pero un método más económico sería la utilización de cámaras fotográficas con pie para realizar tomas fijas. Es decir, si se

[I] `http://lbm.ab.a.u-tokyo.ac.jp/~iwata/shape/index.html`
[II] `http://www.flywings.org.uk/MorphoJ_page.htm`
[III] `http://www.photomodeler.com/`

pudieran adquirir varias fotos desde distintas perspectivas de un mismo objeto, y procesarlas, se podría llegar a un nivel de detalle interesante con un costo mucho más bajo, tanto monetario como de trabajo a la hora de tomar los datos del objeto físico.

En la actualidad el posicionamiento de vértebras, entre otros fósiles, está estrechamente ligado a la mirada del paleontólogo que intenta clasificar ya que no existen métodos formales y automatizados que permitan obtener una aproximación de cuál sera la posición de una pieza en particular. Esto conlleva la desventaja de que el paleontólogo toma como referencia las variaciones que conoce empíricamente, cuando en efecto, si existiera una base de conocimiento común, sería posible evaluar una pieza con mucha mayor precisión y eficacia.

2. Morfometría Geométrica

La Morfometría Geométrica es el conjunto de métodos para la adquisición, procesamiento y análisis de figuras que contienen toda la información geométrica. Definida como la unión entre la biología y la geometría, se ha convertido en una herramienta fundamental para el estudio de estructuras fenotípicas. [Gonzalez 2011]

Su principal innovación teórica radica en un cambio rotundo en la aproximación al tamaño y la forma de las estructuras bajo estudio. En lugar de enfocarse en el análisis multivariante de un conjunto de medidas lineales entre puntos morfométricos, la Morfometría Geométrica propone estudiar los cambios en el tamaño y la forma a partir del desplazamiento en el plano (2D) o en el espacio (3D) de un conjunto de puntos morfométricos o *landmarks*. La relación espacial en dos o tres dimensiones de estos landmarks siempre se conserva a lo largo de todo el análisis, lo que permite reconstruir con tanta precisión como se desee la forma y el tamaño del espécimen estudiado [Gonzalez 2011].

Desde comienzos de los años 90 se ha desarrollado un conjunto de métodos analíticos y gráficos que permiten observar estos cambios espaciales desde una óptica estadística, un paso fundamental en el desarrollo de un método biológico [Gonzalez 2011].

Para entender los fundamentos de la Morfometría Geométrica se hace necesario tener en cuenta conceptos tales como:

- *Landmarks*. Un *landmark* es un punto en un espacio bi o tridimensional que corresponde a la posición de un rasgo en particular en un objeto de interés. [Gonzalez 2011] Los *landmarks* ideales cumplen los siguientes requisitos:
  1. Proveen una adecuada cobertura morfológica.
  2. Pueden ser identificados repetidamente y con precisión.
  3. Deben ubicarse en el mismo plano.
- Contornos/outlines. La forma se captura por medio de las coordenadas de la secuencia de puntos que lo definen. Estos contornos pueden ser cerrados o abiertos, es decir que el comienzo y el fin de un contorno se encuentran o no. Para obtener el contorno de un objeto es posible aplicar diversos métodos:

1. Análisis de Fourier. Se analiza la contribución de los coeficientes de una función trigonométrica que reproduzca lo más exactamente posible una determinada curva. [Gonzalez 2011].
2. Curvas polinomiales. Una ecuación polinómica de grado n ajusta a $n+1$ restricciones, estas restricciones pueden ser puntos, ángulos o una curvatura.
3. Análisis eigenshape.

- Forma y tamaño.

Se debe resaltar que en Morfometría se hace referencia a la forma como los datos que contienen sólo tamaño y contorno (silueta). Si hablamos de shape (silueta), nos referimos a propiedades geométricas de un objeto que son invariantes respecto de la ubicación, escala u orientación. Esta postura da lugar a que:

- Al estar enfocados en las propiedades geométricas podemos dejar de lado propiedades tales como el color y la textura.
- Las cualidades de invariancia respecto de la posición, escala u orientación, se pueden lograr mediante la utilización de medidas invariantes tales como las distancias de radios, ángulos o mediante la utilización de métodos que permiten transformar todos los datos en un sistema de coordenadas común.

Los métodos de Morfometría Geométrica cuantifican la forma (size y shape) de cada espécimen de acuerdo a la ubicación en el espacio de un conjunto de *landmarks* o puntos que son homólogos entre individuos. Luego, el tamaño y la forma son separados a través de la superposición de Procrustes[IV] de los *landmarks*, que traslada dichos puntos a un origen de coordenadas común, los escala a un tamaño en común, y los rota hasta minimizar la suma de cuadrados de las distancias entre sí. Así, la superposición de Procrustes permite cuantificar la forma como una desviación multidimensional de los *landmarks* de un espécimen de una configuración de referencia (usualmente, el promedio de todas las configuraciones de la muestra).

Distintas formas de datos dentro de la Morfometría

Tenemos varias clases de variables para utilizar, éstas se seleccionarán dependiendo de las necesidades a la hora de especificar los puntos a investigar, en nuestro caso hicimos uso de:

- Distancias.
- Contornos.
- *Landmarks* (primarios, secundarios o terciarios).

[IV] El análisis de Procrustes consiste en aplicar una transformación euclideana que conserva la forma del objeto, para eliminar las diferencias de traslación, rotación y escala entre ellas y así llevarlas a un punto estándar para poder ser trabajadas. [Smith 1867]

3. Morfometría y Geometría en Imágenes

Como vimos en la sección anterior, referida a Morfometría Geométrica, para esta propuesta es de máxima importancia reconocer formas y características. Matemáticamente podemos representar la forma como un vector de puntos del tipo (x_i, y_i) (si estamos sobre dos dimensiones). La forma estará almacenada como:

$$Z = [x_1 y_1 x_2 y_2 \cdots x_n y_n] \tag{1}$$

En particular nos interesan las transformaciones que nos permitan obtener información sin perder la forma; éstas pueden ser: rotación, escalado, traslación.

Transformación de Procrustes

La Matriz de Procrustes es una combinación de las tres transformaciones mencionadas; multiplicando estas matrices podemos obtener la misma [Solomon 2011]. La propiedad fundamental de esta matriz es que la multiplicación de cualquier modelo de distribución de puntos por la matriz de Procrustes permite preservar la forma de la misma.
Por lo que la matriz de Procrustes es $P = ERT$,

$$\begin{pmatrix} \alpha & \gamma & \lambda_1 \\ -\gamma & \alpha & \lambda_2 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

donde $\alpha = S\cos\theta$, $\lambda_1 = S(\beta_x \cos\theta + \beta_y \sin\theta)$, $\beta = S\sin\theta$ y $\lambda_2 = S(-\beta_x \sin\theta + \beta_y \cos\theta)$

Procrustes es utilizado para trasladar la forma entrante a un punto neutro (acordado), escalar la imagen a un tamaño ya existente y rotarla para que concuerde con la forma que se tiene como plantilla, para que esto quede clarificado se muestra la idea en la Figura 2.
Esto nos permitió a la hora de manipular las imágenes de vértebras llevarlas a un espacio común sin perder información necesaria para su posterior clasificación.

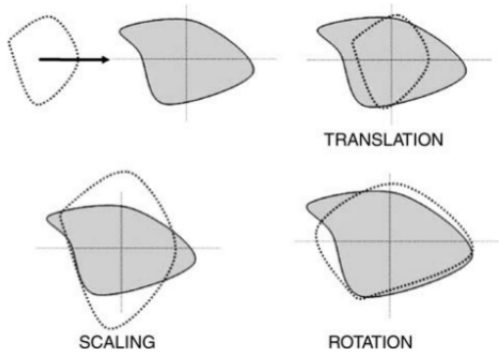

Figura 2: Transformación de Procrustes [Solomon 2011].

Operaciones Morfológicas

La dilatación no es una transformación, sino una operación no lineal, no inversible. El filtrado morfológico se basa en asumir que la imagen a procesar puede descomponerse binariamente en dos componentes, figura y fondo. También existe un elemento estructurante, el cual, de acuerdo a cuál sea la operación morfológica, se utiliza para alterar la figura o el fondo.
La dilatación consiste en apoyar el elemento estructural sobre cada punto dentro la figura, y todos los nuevos puntos alcanzados (que antes estaban en el fondo) pasan ahora a la figura. En la Figura 3 se muestra esta operación aplicada a nuestra investigación en particular.
La erosión consiste en apoyar el elemento estructural sobre cada punto fuera de la figura, y todos los puntos alcanzados (que antes estaban en la figura) pasan ahora al fondo.

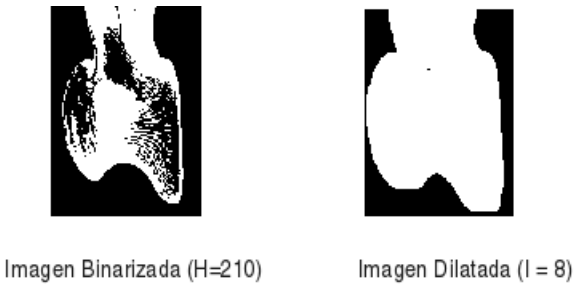

Figura 3: Ejemplo de una iteración de dilataciones sobre el cuerpo espinal de una vértebra dorsal.

Gradiente Morfológico

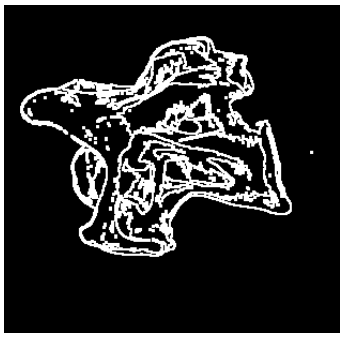

Figura 4: Gradiente morfológico de una vértebra cervical.

El gradiente morfológico se puede definir fácilmente como la diferencia entre la dilatación y la erosión. La morfología puede extenderse a imágenes en escala de grises, tomando la máxima luminancia de los pixels visitados por el elemento estructural para la dilatación, y la mínima para la erosión. Además, hay varias maneras de definir el gradiente morfológico. La imagen dilatada menos la original da el borde exterior en figuras blancas sobre fondo negro y el interior si la figura es negra sobre fondo blanco. La imagen original menos la erosionada da los casos duales. Sea b un elemento estructurante e i una imagen en escala de grises, se llama gradiente morfológico a:

$$G(i) = (i \oplus b) - (i \ominus b) \tag{3}$$

Estas operaciones fueron utilizadas para eliminar detalles innecesarios que podrían traer errores a la hora de tomar los datos para la clasificación, un ejemplo puede de Gradiente Morfológico puede observarse en 4.

Segmentación

En términos simples podemos decir que la segmentación nos permite fraccionar una imagen en partes significativas para nuestro problema. Una separación clásica es dividir nuestro objeto de estudio del fondo, pero nuevamente debemos resaltar que la división de nuestra imagen se especifica en función del problema a resolver. La rama de segmentación se divide es dos líneas básicas:

- Contornos.
- Regiones.

Un ejemplo clásico y simple es utilizar métodos de binarización, los cuales nos pueden ayudar tanto para diferenciar nuestro objeto del fondo, como para facilitar la búsqueda de contornos.

Contornos

En especial veremos el algoritmo de búsqueda de contornos propuesto por Canny que fue el utilizado en *PyBones* como puede observarse en la Figura 5. Básicamente lo que busca es reducir la cantidad de datos que se poseen de una imagen manteniendo la idea estructural inicial de la misma.

Con el algoritmo se desea optimizar:

- Detección, manejar en forma óptima las probabilidades de que el contorno encontrado sea real.
- Localización, el contorno encontrado debe estar lo más cerca posible del contorno real.
- Número de resultados (number of responses), se deben corresponder uno a uno entre contornos reales y encontrados.

El algoritmo consiste en [Canny 2009]:

1. Se suaviza la imagen utilizando un kernel Gaussiano.
2. Se encuentra la intensidad del contorno mediante un operador de Sobel, que toma el valor del gradiente (éste nos da la intensidad del borde) en las direcciones horizontales y verticales. $E(x, y) = G_x(x, y) + G_y(x, y)$
3. Se calcula la dirección del borde, mediante su ángulo. $\theta = \frac{G_x(x,y)}{G_y(x,y)}$
4. Se digitaliza el ángulo obtenido, esto es aproximar la dirección digitalmente.
5. Se suprimen los no-máximos para delinear el borde.
6. Histéresis, se binariza la imagen para detectar los pixels pertenecientes al borde con dos umbrales, uno mínimo y otro máximo.
   - Si $E(x, y) < T_1$, se rechaza el pixel y no forma parte del borde.
   - Si $E(x, y) > T_2$, se acepta el pixel y forma parte del borde.
   - Si $T_1 < E(x, y) < T_2$, se acepta sólo si éste es conector de pixels que determinan el borde.

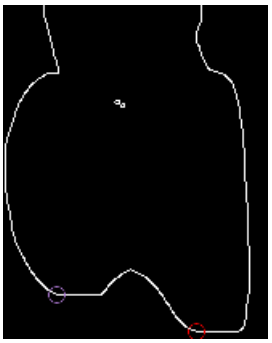

Figura 5: Detección de bordes mediante el Algoritmo de Canny en el cuerpo espinal de una vértebra dorsal.

4. Agente Clasificador: Redes Bayesianas vs Redes Neuronales

Una Red Bayesiana es un grafo acíclico dirigido, en el que los nodos representan variables aleatorias y los arcos representan influencias causales. Esta estructura nos permite representar e inferir sobre un dominio incierto.

Sea $U = X_1, X_2, ..., X_n$ un conjunto de variables aleatorias. Una red bayesiana para U es una dupla $B =< G, T >$ en la que:

1. G es un grafo acíclico dirigido en el que cada nodo representa una de las variables $X_1, X_2, ..., X_n$, y cada arco representa relaciones de dependencia directas entre las variables. La dirección de los arcos indica que la variable de llegada depende de la variable situada en su origen.

2. T es un conjunto de parámetros que cuantifica la red. Contiene las probabilidades $P(x_i|_i)$ para cada posible valor x_i de cada variable X_i y cada posible valor Πx_i *de* X_i, donde este último denota al conjunto de padres de X_i en G.

Los estados que puede tener una variable (nodo) deben cumplir las siguientes propiedades [Britos 2005]:

- Un nodo puede encontrarse sólo en uno de sus estados en un momento t.
- Un nodo no puede tener ningún valor fuera de su conjunto de valores posibles.

Redes Neuronales

Para introducir el concepto de Redes Neuronales Artificiales debemos rever cómo están compuestas las neuronas y cuál es su función. La neurona es una célula viva, está compuesta por un cuerpo neural, un tronco principal llamado axón y pequeñas ramificaciones llamadas dendritas. Una característica primordial de las neuronas es su capacidad para comunicarse generando señales que pueden ser químicas como eléctricas. En términos simples una neurona

recibe una señal de otra neurona, la procesa en su cuerpo y emite una señal de salida.

Sus componentes son:

1. Nodos o Neuronas, son elementos básicos de una red neuronal con los que se pueden generar las representaciones que deseemos. Las neuronas se clasifican en:
   - Entrada, reciben los estímulos provenientes del exterior, es decir, desde afuera de la red.
   - Internas o unidades ocultas, aquéllas cuyas entradas y salidas se encuentran dentro de la red.
   - Salida, son las que muestran el resultante del procesamiento de las entradas en la red.
2. Conexión, por ésta viaja la señal que permite comunicar a las neuronas entre sí. Cada canal tiene un peso asociado que normalmente es multiplicado a la señal enviada, estos canales suelen ser unidireccionales.
3. Modelo de actualización de la red, dependiendo del caso puede ser síncrono o asíncrono.

4.1. Análisis: ¿Redes Bayesianas o Redes Neuronales?

Si bien las redes del tipo Naive Bayes [V]son ampliamente utilizadas para clasificación, a la hora de medir procesamiento de máquina quizás sea oportuna la búsqueda de otro método menos exigente en complejidad de cálculos. A grandes rasgos, en relación con el cálculo de la probabilidad de cada nodo respecto de sus padres, contando también los cálculos de generar las tablas con todas las variables de estado que puede tener un nodo.

Las Redes Bayesianas son recomendables cuando su estructura es no lineal ya que estas muestran de forma sencilla para el humano ya que la causalidad es mucho más visible que en las Redes Neuronales, pero al estar trabajando previamente con procesamiento de imágenes en nuestro desarrollo el tiempo de cómputo es algo para tener en cuenta y mientras más complejo es el trabajo más prolongado será el tiempo de retardo.

Al ser nuestro problema del tipo lineal y nuestras entradas a clasificar básicamente vectores, podemos considerar el uso de otros métodos como pueden ser las Redes Neuronales, no sólo teniendo en cuenta que las matrices se reducen notablemente sino que los cálculos se restringen a sumatorias y multiplicaciones. Además, podemos tener en cuenta que en el caso de nuestras variables no es primordial tener una probabilidad de su existencia, sino que es suficiente decir si se encuentran o no y en qué posición.

Otro aspecto interesante por el cual inclinarse hacia Redes Neuronales es su capacidad de aprendizaje adaptativo. Esto significa que aprenden a llevar a cabo ciertas tareas mediante un entrenamiento con ejemplos ilustrativos. Como las Redes Neuronales pueden aprender a diferenciar patrones mediante ejemplos y entrenamientos, no es necesario elaborar modelos a priori ni necesidad de especificar funciones de distribución de probabilidad como en las Redes

[V]Es un clasificador probabilístico basado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales. Es a causa de estas simplificaciones, que se suelen resumir en la hipótesis de independencia entre las variables predictoras, que recibe el apelativo de ingenuo (Naive).

Bayesianas. La capacidad de generalización de las Redes Neuronales, cuando las posibles combinaciones de patrones de entrada son tantas que resultaría imposible especificarle a un dispositivo qué hacer en cada caso, es un punto importante puesto que la red se entrena con un número de patrones representativo y no con la totalidad de ellos. [Britos 2005]

Por lo citado, durante el desarrollo de este trabajo se ha avanzado en la investigación más allá de las Redes Bayesianas, y además de un modelo de agente clasificador basado en Redes Bayesianas se ha definido la implementación de la aplicación PyBones, que optimiza el proceso desde el punto de vista de performance, aplicando Redes Neuronales.

5. Resultados

El trabajo desarrollado hasta el momento no sólo ha logrado una propuesta de aplicación de *landmarks* y Redes Bayesianas para asistir al paleontólogo en el posicionamiento de vértebras sino que, así mismo, mediante Redes Neuronales, se logró desarrollar un agente que a partir de imágenes puede discriminar inteligentemente a qué posición pertenece una determinada vértebra con un cierto valor de confianza.

PyBones, la aplicación resultado de este proyecto, que se muestra en la Figura 5 fue creada en su totalidad con Software Libre[VI] y la aplicación resultante mantiene las mismas políticas; Permite:

1. Cargar imágenes (una o varias) para ser analizadas.
2. Modificar valores de entrenamiento para la red.
3. Entrenar a la red neuronal que evalúa las vértebras.
4. Mostrar los resultados obtenidos por la red en forma gráfica.
5. Guardar los resultados en formato pdf.
6. Contar con una base de datos de las vértebras estudiadas sin necesidad de almacenar las imágenes.

Además se remarca que para la clasificación no es necesario almacenar imágenes anteriores, sólo basta con retener valores específicos, tales como posiciones y distancias.

En la Figura 7 se muestran los resultados del posicionamiento de las 23 vértebras correspondientes a un Saurópodo mediante la aplicación desarrollada, PyBones y un conjunto de alrededor de 40 imagenes de vértebras que fueron proporcionadas por Investigadores del *Museo Egidio Feruglio*[VII], y se deja constancia del grado de confianza proporcionado por un humano[VIII] en relación con los valores estimados por la red y el límite de confianza.

Si bien *PyBones* enfoca la clasificación de vértebras en particular, se destaca que se ha

[VI] Según la *Free Software Foundation*, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.
[VII] `http://mef.org.ar/`
[VIII] Estos valores se tomaron de forma empírica junto con el paleontólogo.

Figura 6: PyBones muestra la posición aproximada de las vértebras analizadas.

logrado definir un esquema de desarrollo y múltiples herramientas reutilizables para la clasificación de otros fósiles o elementos que generan las mismas dificultades a la hora de ser clasificados, como por ejemplo los dientes, ya que los *landmarks* y variaciones serán diferentes pero el sistema de almacenamiento de éstos y la utilización de la red neuronal para clasificarlos serán los mismos.

Es decir, se estableció un agente en el cual sólo se debe cambiar el elemento (y *landmarks*) a examinar y podrá ser reutilizado.

Además, con *PyBones* se logró tener una base de datos con información de vértebras sin tener la necesidad de almacenar las imágenes reduciendo así el costo de espacio, en lugar de utilizar megas por cada imagen se utilizan unos pocos bytes de números enteros. Más allá de otras mejoras y/o extensiones que podrían tenerse en cuenta y detallarse, en un futuro inmediato sería muy valioso poder aplicar el mismo procedimiento trabajando con más de una vista de los elementos a clasificar, de tal manera de contar con mayor cantidad de información. Esto implicaría tener que reconocer un mismo landmark en varias perspectivas haciendo uso de planos epipolares.

Otra mejora destacable, considerada de valor a corto plazo, y en gran parte ya prevista durante el desarrollo de la aplicación, es implementar de un modo conectar/desconectar (plug/unplug) la inserción de distintos fósiles u otros objetos a clasificar, como es el caso de dientes, por ejemplo.

También se deja para un análisis futuro otro tipo de agentes clasificadores, como pueden ser Máquinas de Vectores de Soporte, Árboles de Decisión, etc.

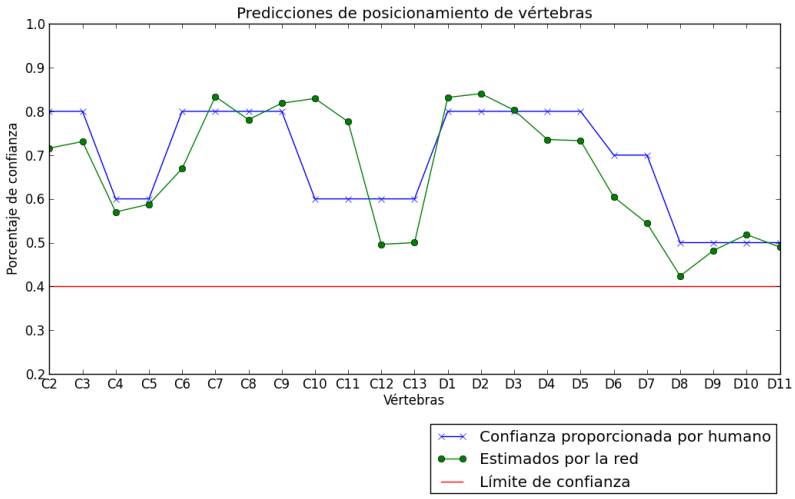

Figura 7: Resultados del posicionamiento de vértebras mediante *PyBones*.

A. Implementación de *PyBones*

A continuación detallaremos las herramientas/bibliotecas utilizadas en el diseño, desarrollo y testeo de *PyBones*.

Lenguaje de Programación El lenguaje de programación elegido fue *Python 2.7.2* ya que es un lenguaje muy flexible, sintacticamente limpio y con una gran comunidad de fondo, ademas cuenta con el soporte de varías bibliotecas que fueron utilizadas y se detallan más adelante.

Entorno de Trabajo Para la codificación y realización de pruebas se utilizó *Prymatex*[IX] un editor basado en PyQt, que soporta Bundles de TextMate, contiene Snippets, resaltado de sintaxis, entre otras bondades.
Para el diseño de la intefaz gráfica se utilizó *QT Designer* es una herramienta que permite crear y construir la interfaz gráfica mediante componentes de Qt de la forma WYSIWYG [X].

Almacenamiento Persistente Para el almacenamiento de los *landmarks* se utilizó ZODB Zope Object Database[XI]; es una base de datos orientada a objetos para almacenar de forma transparente y persistente objetos en el lenguaje de programación Python. Se incluye como parte de Zope, un Servidor de aplicaciones Web, que también puede ser

[IX] `http://github.com/organizations/prymatex`
[X] what-you-see-is-what-you-get
[XI] `http://www.zodb.org/`

utilizado independientemente de Zope.

Control de Versiones Para el control de versiones se utilizó *Git*[XII]; Git es un sistema de control de versiones diseñado para manejar proyectos muy grandes con velocidad y eficiencia, pero igual de apropiado para repositorios pequeños. Corresponde a la categoría de herramientas de manejo de código fuente distribuído, similar por ejemplo a Mercurial o Bazaar.

Las bibliotecas usadas en el desarrollo de *PyBones* fueron:

Python-Perceptron [XIII] Esta biblioteca implementa un perceptron mediante *NumPy*, especialmente para nuestra aplicación se modificaron algunas líneas de código que se encuentran en el repositorio de *PyBones*.

PyQt [XIV] Es un conjunto de bindings para Python del framework Qt de Nokia; este corre en todas las plataformas Qt, Windows, MacOS/X and Linux.

Xutils [XV] Genera hojas de cálculo compatibles con OpenOffice.org Calc, LibreOffice Calc, Gnumeric y Excel 97/2000/XP/2003.

ReportLab [XVI] Es un módulo que provee todo lo necesario para generar reportes de calidad de presentación profesional desde Python.

PIL [XVII] Otorga soporte para muchos formatos de archivos y procesamiento de imágenes en general.

NumPy [XVIII] Permite manipular de manera rápida y eficiente arreglos numéricos tales como vectores y matrices (pero también arreglos multidimensionales de rango arbitrario).

OpenCV [XIX] OpenCV (Open Source Computer Vision) es una biblioteca de funciones para la vision artificial en tiempo real.

SimpleCV [XX] SimpleCV es una interface a varias bibliotecas de código abierto sobre visión artificial en un solo paquete.

[XII] `http://git-scm.com/`
[XIII] `http://code.google.com/p/python-perceptron/`
[XIV] `http://www.riverbankcomputing.co.uk/software/pyqt`
[XV] `http://www.simplistix.co.uk/software/python/xlutils`
[XVI] `http://www.reportlab.com/software/opensource/`
[XVII] `http://www.pythonware.com/products/pil/`
[XVIII] `http://numpy.scipy.org/`
[XIX] `http://opencv.willowgarage.com/wiki/`
[XX] `http://simplecv.org/`

También otorga cierta abstracción de detalles a la hora de usar los algoritmos clásicos.

A.1. Determinación de *Landmarks* a Utilizar

Para tomar los distintos *landmarks* lo primero que se realizó fue la división en *Regiones de Interés*, las que se pueden ver detalladas en la Figura 8. Luego, tomando cada región en forma independiente, se aplicaron transformaciones que permiten obtener los datos necesarios para posicionar los *landmarks* indicados por el experto.

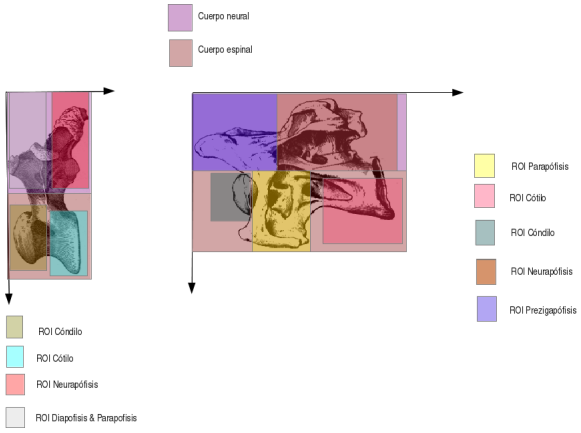

Figura 8: ROI de las vértebras (Dorsales y Cervicales).

A.1.1. *Landmarks* sobre Vértebras

Para la obtención de los *landmarks* se trabaja con imágenes cargadas en forma de escala de grises y luego binarizadas y en especial para las vértebras dorsales se trabaja con dilatación y obtención de contornos.

Luego se regionalizan las secciones de interés como se muestra en el gráfico 8 y se buscan ciertas características o formas. Las Figuras 9, 10 y 11 muestran cómo se marcaron las imágenes luego de ser analizadas para asegurar que los puntos encontrados fuesen los correctos.

Una vez obtenidas las posiciones es posible aplicar diferentes funciones, como por ejemplo medir la distancia (con ejes cartesianos) entre la diapófisis y la parapófisis.

A.2. Almacenamiento de *Landmarks* y Distancias

Dado que la implementación se soporta sobre el modelo de objetos, se definió el uso de una Base de Datos Orientada a Objetos, en especial ZODB (Zope Object Database).

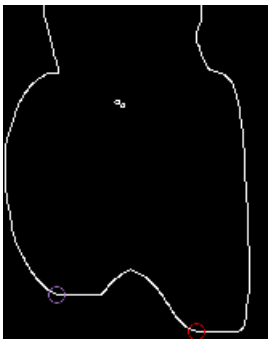

Figura 9: Posición del cótilo y el cóndilo.

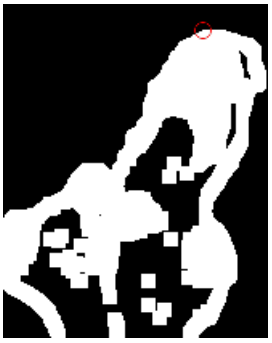

Figura 10: Posición de la neurapófisis.

La mayor ventaja que otorga ZODB es la transparencia, sólo se debe asegurar que los objetos que generamos son persistentes, luego su función es similar a la de un diccionario en Python.

Entre otras de sus bondades ZODB permite transacciones, capacidades de deshacer, almacenamiento conectable de forma transparente (a bases relacionales, repositorios remotos, etc.), almacenamiento en Cache, etc.

Debemos tener en cuenta que las clases que deseamos almacenar deben heredar de *Persistent* y como se ve en la Figura 12 la clase *Dorsal* hereda de *Vertebra*:

```
import ZODB
from Persistence import Persistent
from uuid import uuid4

class Vertebra(Persistent):
    """La clase Vertebra ser  la clase padre para los dos tipos
     de vertebras (cervicales y dorsales)"""

def __init__(self):
```

Figura 11: Posición de la diapófisis y parapófisis.

```
        """Inicializa con un id para la db y entradas para la red"""

        random_id = uuid4()
        self.id = random_id.int
        self.dcotidilo = None
        self.dparadia  = None
        self.pneurapof= None
        ....
        ....
        ....
```

Y a continuación se muestra la implementación de la clase *Dorsal* que es muy similar a la clase *Cervical*.

```
class Dorsal(Vertebra):
    """La clase dorsal agrupa ambas zonas de las vertebras dorsales
    neural y espinal"""

    def __init__(self,d_neural,d_espinal):
        """guarda el cuerpo neural y espinal previamente creados"""
        super(Dorsal, self).__init__()
        self.neural = d_neural
        self.espinal = d_espinal
        . . . . . .
        . . . . . .
        . . . . . .
        . . . . . .
```

PyBones almacena valores, posiciones de los distintos *landmarks* en la base de datos, por lo que se descarta almacenar el peso de la imagen procesada. La clase que ZODB almacena es *Vertebra*.

A.3. Diseño del Pipe de Procesamiento de Imágenes

En el pipe de procesamiento definimos cuáles son las operaciones que se aplican a las imágenes para obtener los datos pertinentes a la tesina; también se definen en el mismo los

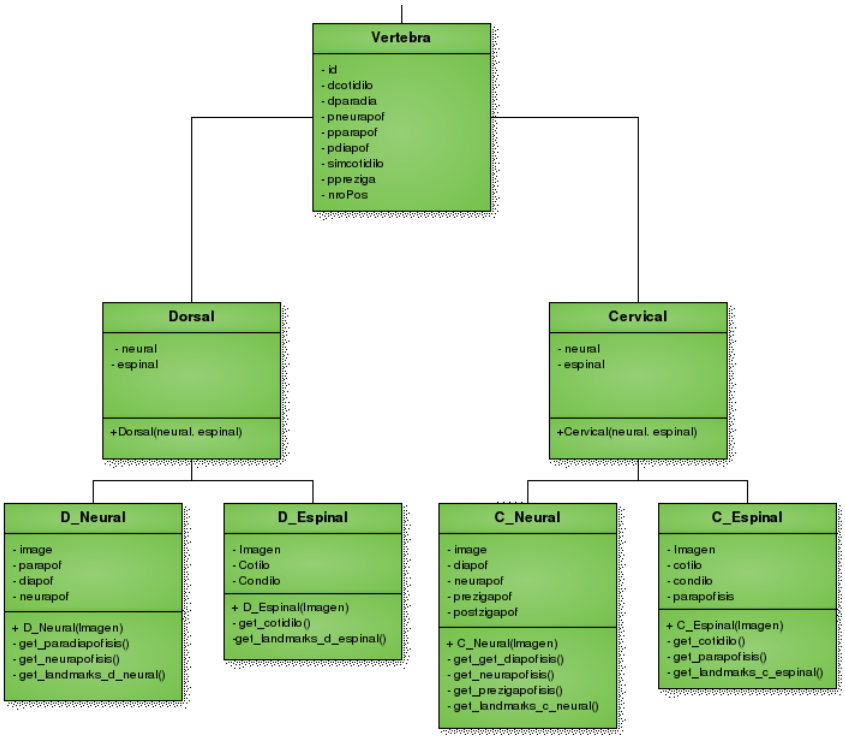

Figura 12: Clase Vertebra y sus subclases

valores necesarios para cada operación y en qué momento se almacenan los resultados.

En la Figura 13 se muestra a modo de ejemplo el pipe implementado para la obtención de *landmarks* de las vértebras dorsales.

A.4. Diseño de la Red Neuronal

Para simplificar las cuestiones de rangos se desarrolló una red neuronal con nodos binarios, esto quiere decir que sus valores pueden ser 1s o 0s.

Como toda elección tienen sus ventajas y desventajas, al ser más sencilla la delimitación de rangos, las redes con neuronas binarias suelen ser más numerosas, dado que por ejemplo, si tenemos una variable, distancia entre Cótilo y Cóndilo, y sus distintos valores son pequeño, mediano, grande, muy grande, se necesitarán 4 neuronas que tomen los respectivos valores de (0001, 0010, 0100, 1000).

Es así que, dadas la variables de este caso, son necesarias 35 neuronas de entrada.

Las variables tomadas en cuenta para este problema son:

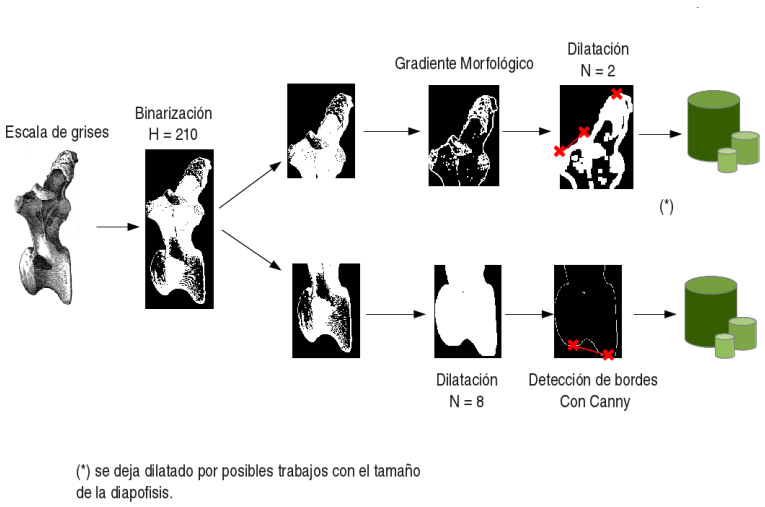

Figura 13: Pipe de obtención de datos espinales.

1. Distancia entre Cótilo y Cóndilo.→ [pequeño, mediano, grande, muy grande]
2. Distancia entre Diapófis y Parapófisis →[inexistente, muy pequeño, pequeño, mediano, grande, muy grande]
3. Posición de la Neurapófisis → [bajo, mediano, alto, muy alto]
4. Posición de la Prezigapófisis →[inexistente, muy pequeño, pequeño, mediano, grande, muy grande]
5. Posición de la Parapófisis →[inexistente, muy pequeño, pequeño, mediano, grande, muy grande]
6. Posición de la Diapófis→[inexistente, muy pequeño, pequeño, mediano, grande, muy grande]
7. Simetría entre posición de Cótilo y Cóndilo →[simétrico, hacia la derecha, hacia la izquierda]

Cada una de éstas con sus correspondientes valores binarios. A modo de ejemplo se muestra una matriz ejemplo 4 de valores para la variable de *Posición de la Neurapófisis* y se destaca que para cada variable se modifica la cantidad de 0s, dependiendo de la cantidad de posibles valores, y que el mecanismo es el mismo.

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{4}$$

A.4.1. Entrenando a la Red Neuronal

El aprendizaje implementado en *PyBones* es del tipo supervisado y off-line, es decir que el tipo de entrenamiento que se le dá a la red es un conjunto de ejemplos de entrenamiento antes de darle un elemento a clasificar, y ésta, en función de lo aprendido tratará de clasificar el nuevo objeto.

Para evaluar su rendimiento se realizaron pruebas en forma individual antes de ser implementado, una de ellas se muestra a continuación.

```
#! /usr/bin/python2
# -*- coding: utf-8 -*-

import perceptron
import xlrd
import numpy as np
from pylab import *


def cargarPatronEntrenamiento(path):

        book = xlrd.open_workbook(path)
        sheet = book.sheet_by_index(0)
        patron = {}
        entrada = []

        for i in range(1,sheet.nrows):
            entrada = []
            for j in range(1,sheet.ncols):
                entrada.append([sheet.cell_value(i,j)])
            patron[i-1]= (np.array(entrada))
           # print patron[i]
        return patron

def entrenar(network, patron):
        """Con la matriz de entrenamiento proporcionada entrenar ,
        modificar la matriz de pesos"""

        for j in range(50): #TODO ...
            for i in patron.iterkeys():
                network.learn(patron[i], i)
                print "entrada = ",patron[i]
                print "salida = ",i
                print type(patron[i])

if __name__ == "__main__":
        per = perceptron.Perceptron(35,23)
        patron = cargarPatronEntrenamiento("../data/entrena.xls")
```

```
        entrenar(per, patron)
        forPlot = []
        for i in patron.iterkeys():
              output = per.classify(patron[i])
              print output[1]
              forPlot.append(float(output[1]))
        ylim(0.20,1.0)
        xticks( arange(0,23), ('C2', 'C3', 'C4', 'C5', 'C6', 'C7',
                               'C8', 'C9', 'C10', 'C11', 'C12',
                               'C13', 'D1', 'D2', 'D3', 'D4', 'D5',
                               'D6', 'D7', 'D8', 'D9', 'D10',
                               'D11') )
        plot(arange(0,23),[0.80,0.80,0.60,0.60,0.80,0.80,
                           0.80,0.80,0.60,0.60,0.6,0.60,
                           0.80,0.80,0.80,0.80,0.80,0.70,
                           0.70,0.50,0.50,0.50,0.50,],'x-',
                           label='Confianza proporcionada por humano')

        plot(arange(0,23),forPlot,'o-',label='Estimados por la red' )

        y = []
        for i in range(23):
            y.append(0.40)

        plot(arange(0,23),y,'-',label=u'L mite de confianza')

        xlabel(u'V rtebras')
        ylabel('Porcentaje de confianza')
        title(u'Predicciones de posicionamiento de v rtebras')
        legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
        show()
```

A.5. Diagrama de Clases

En esta sección se analizan exclusivamente las clases fundamentales de PyBones.

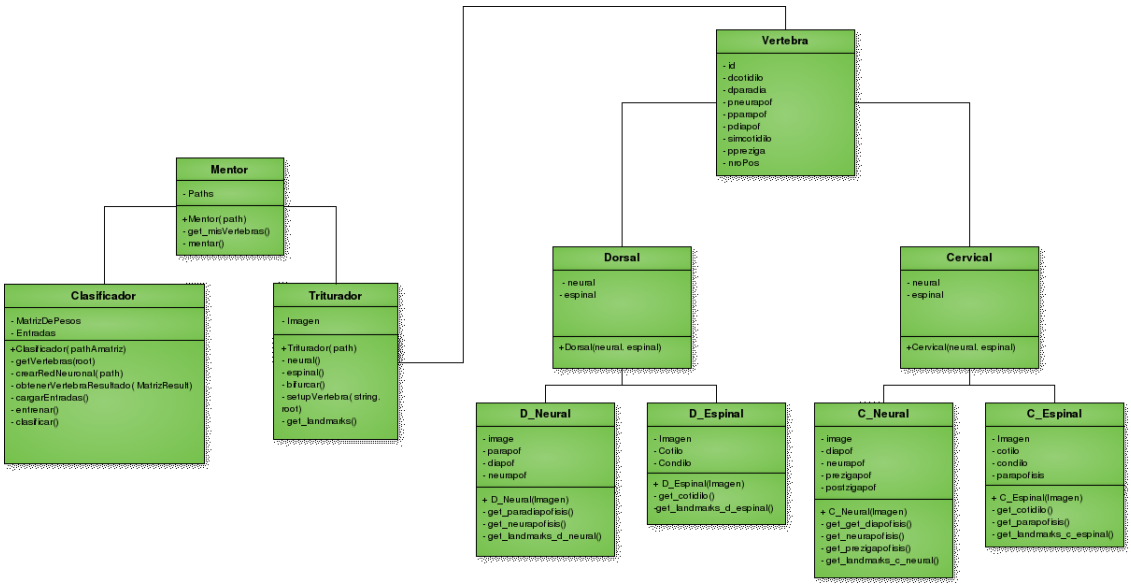

Figura 14: Diagrama de Clases

Como se observa en la Figura 14 la lógica e inteligencia de la aplicación se encuentra bien diferenciada del objeto a analizar, lo que nos permitirá en casos futuros poder modificar el elemento a evaluar manteniendo la lógica intacta.

A.5.1. Clases del núcleo de *PyBones*

- **Mentor**, es el encargado de administrar los Trituradores de las n imágenes y su Clasificador.
- **Triturador**, es la encargada de preparar la imagen ingresada para la adquisición de *landmarks*, separando las regiones a estudiar y aplicando las respectivas transformaciones. También es la encargada de almacenar en la base los datos obtenidos luego del análisis.
- **Clasificador**, es la encargada de contener a la red neuronal, entrenarla, obtener de la base de datos las entradas para la misma y mostrar los resultados.
- **Vertebra**, es la que contiene los *landmarks* obtenidos por el Triturador.
  Se destaca particularmente que se incluye en el núcleo dado que representa los objetos a

analizar, requeridos para que *PyBones* cumpla su función, si bien sería posible proveer otra clase de objetos a *PyBones* para su clasificación.

Referencias

[Fergus 2007] Weakly Supervised Scale-Invariant Learning of Models for Visual Recognition.R. Fergus,P. Perona,A. Zisserman, INTERNATIONAL JOURNAL OF COMPUTER VISION Volume 71, Number 3 (2007), 273-303, DOI: 10.1007/s11263-006-8707-x, Springer 2007.

[Steger 2001] Occlusion, Clutter, and Illumination Invariant Object Recognition. Carsten Steger, PATTERN RECOGNITION Lecture Notes in Computer Science, 2001, Volume 2191/2001, 148-154, DOI: 10.1007/3-540-45404-7_20, Springer 2001.

[Szeliski 2010] Computer Vision:Algorithms and Applications. Richard Szeliski,Springer 2010. pp 655-731.

[Felgaer 2005] Optimización de Redes Bayesianas basado en Técnicas de Aprendizaje por Inducción. Pablo Ezequiel Felgaer, UBA 2005.

[Larrain 2002] Clasificación Bayesiana, Tópicos Avanzados en Base de Datos. Carlos Hurtado Larrain, Universidad de Chile 2002.

[Ramoni 1997] Efficient Parameter Learning in Bayesian Networks from Incomplete Databases. M. Ramoni, P. Sabastiani, Technical Report KMi-TR-41, Knowledge Media Institute, The Open University, January 1997.

[Mery 2002] Visión Artificial. Domingo Mery, 2002.pp 85-99.

[Forcada 2003] Clasificación Supervisada basada en Redes Bayesianas. aplicación en Biología Computacional.Victor Robles Forcada,ID 953, Archivo Digital UPM 2003.

[Villanueva 2007] Inteligencia Artificial II.David A. Velasco Villanueva, 2007.

[Juarez 2008] SEDFE: Un Sistema Experto para el Diagnóstico Fitosanitario del Espárrago usando Redes Bayesianas. Pedro Shiguihara Jurez, Jorge Valverde Rebaza,2008.

[Svarney 2010] The Handy Dinosaur Answer Book.Patricia Barnes-Svarney and Thomas E. Svarney, Visible Ink Press, 2 edition 2010.

[Osborn 1921] Osborn Mook 21 - Camarasaurus Amphicoelias and other cope sauropods ,1921.

[Gonzalez 2011] Introducción a la Morfometría Geométrica (Apuntes de curso), Silvina Van der Molen, Rolando González, 2011.

[Slice] Chapter One, Modern Morphometrics, Dennis E. Slice,ISBN-10: 0306486970, Springer 2005.

[Smith 1867] Smith, W., ed. (1867), Procrustes, A Dictionary of Greek and Roman biography and mythology, Boston: Little, Brown Co.

[Foster 2007] Foster, John (2007). Jurassic West: The Dinosaurs of the Morrison Formation and Their World. Indiana University Press, 201, 248. ISBN 978-0-253-34870-8.

[Britos 2005] Britos, Hossian, García Martinez, Sierra, Mineria de Datos, ISBN 987-1104-30-8, Editorial Nueva Librería 2005.

[Neu2011] Redes de Neuronas (Grupo 2 - RAI - UC3M 2010/2011)

[Nano 2008] Blog de Nanotecnología, Marzo 2012.

[Canny 2009] Canny Edge Detection 09gr820 March 23, 2009.

[Morfo 2012] Operaciones Morfológicas, Febrero 2012

[Solomon 2011] Fundamentals of Digital Image Processing, A Practical Approach with Examples in Matlab.Chris Solomon & Toby Breckon, ISBN-10: 0470844728, Wiley 2011.