TEMA 9. Redes Neuronales.

Neuronas.

- Son dispositivos de "**todo o nada**".
- Las **salidas** que provoca pueden **afectar** al **resultado** final.
- Si en la neurona **biológica** las entradas son **dendritas**, en la **neurona computacional** las entradas son **número reales**.
- La **inicialización** de **pesos** en una red neuronal debe realizarse de forma **arbitraria**.
- El modelo computacional **se asemeja** a una neurona biológica.
- Una neurona puede decir si va a llover o no (**elección si o no**).
- En una computacional la integración es la **suma** ponderada(net) por los pesos sinápticos seguida de **una** función de **activación** f(**net**).
- Para que una neurona dispare salida es necesario **superar** un umbral.

Dendritas, nº reales; Axón, suma ponderada; **Soma, resultado**. Siendo las primeras de cada pareja las partes de una neurona biológica y la segunda las de una neurona computacional.

Neuronas artificiales (Red neuronal).

Se basan en el **aprendizaje supervisado**.
Converge cuando **el error de validación se mantiene bajo** y los ejemplos de entrenamiento **no** provocan cambios significativos en los pesos de la red.
Puede sufrir sobre entrenamiento y arrojar resultados erróneos debido al **exceso de flexibilidad o rigidez**.
El sobre entrenamiento se puede detectar cuando el error de **validación sube** después de haber alcanzado niveles mínimos. El tiempo de entrenamiento es lento, y el tiempo de respuesta una vez entrenada puede ser lento.

Multicapa.

- Con perceptrones multicapa si no se produce procesamiento nos encontramos en la **capa de entrada**.
- El error δ: **una neurona** de una capa intermedia **contribuye** en los δ de la **capa siguiente**.
- No usar entrenamiento multicapa puede provocar el problema de la no-separabilidad lineal.
- Para resolver **problemas de paridad** es preciso **incorporar una capa adicional** de neuronas.
- Se buscan **funciones derivables** con forma **similar** al escalón del **perceptrón** de **una** sola **capa.**
- Se puede aplicar el algoritmo de **entrenamiento multicapa** si la función de activación es **derivable**.

Interpretación geométrica.

El ajuste de hiperplanos se forma, dado **dos conjuntos** de ejemplos correspondientes a dos clases, buscaremos su separación por un hiperplano.
Un perceptrón puede aproximar cualquier función si **posee dos capas ocultas**.
La interpretación geométrica de la función de activación "umbral" es un **hiperplano**.

Backpropagation.

- Itera hasta que el error **baje** de un umbral.
- Es necesaria una **función derivable**.
- En la convergencia con valores **muy pequeños** hay **convergencia lenta**.
- Las fases son: **Entrada, integración y salida**. (Neuronas **biológicas y computacional)**

Mide lo que **contribuye cada neurona** al error obtenido: $\delta_k = (d_k - y_k) * f'(net_k)$

Aprendizaje estocástico (Gradiente).

La inicialización de los pesos es **arbitraria y aleatoria** y el problema de la inicialización en los descensos por gradiente **tiene** solución. (D**istintas** inicializaciones)
En una inicialización de red Arbitraria y Aleatoria, son los **mínimos locales**.
La solución aportada a dicho problema es **entrenar la red** desde las distintas inicializaciones.

Ajuste de la red.

El **overshooting** ocurre cuando nos **saltamos el máximo**.
En la **inicialización de red** se pretende **evitar** el **mínimo local**.
El problema XOR es un problema de **paridad**.

Regla Delta.

- **Permite ajustar** iterativamente el hiperplano/**plano**.
- Se asume que el **incremento** de los pesos es proporcional a la disparidad entre la **salida** observada y la salida deseada.
- Se utiliza en el aprendizaje **supervisado**.
- Si el peso utilizando la regla delta es demasiado grande se **reduce** el peso de las **aristas**.
- En el entrenamiento de perceptrones de una neurona si todos los ejemplos están bien etiquetados después de un número de iteraciones diremos que los **conjuntos** de ejemplos son **linealmente separables**.

En la interpretación geométrica es **falso** que los **problemas** con regiones de decisión más **complejas no requieren** distintas **estrategias** de **separación**.
En los perceptores multi-capa es **falso** que la "interpretación geométrica" se desarrolló sobre un algoritmo que **permite encontrar los pesos asociados a todas las neuronas**.

TEMA 10. Boosting y Adaboost.

Un poco de notación.

Están compuestos por: Patrones, Clases, Conjunto de entrenamiento**, función aprendida,** clasificador**.**

Combinar clasificadores "débiles".

- Los clasificadores débiles son moderadamente **precisos**, **simples** y **funcionan** al menos **mejor** que una clasificación aleatoria.
- Los **árboles** de **decisión** y las **redes neuronales** son **clasificadores débiles**/**inestables**.
- A más clasificadores que se añadan **no** se mejorará el aprendizaje de datos.
- Si **combinamos** un conjunto de **clasificadores** obtendremos **uno** más **fuerte** que los mismos por separado.
- Combinando **muchos** clasificadores **débiles** obtendremos un **clasificador fuerte**.

Boosting vs Bagging.

Un **clasificador** más **fuerte** se puede formar por el conjunto de Votos ponderados (**clasificadores**) y Muestreo ponderado (**ejemplos**). Los **clasificadores ponderados** se utilizan en los clasificadores con el **mismo peso** en el voto.

- **Boosting no** combina los clasificadores con el mismo peso en el voto.
- En boosting cada nuevo modelo está **influenciado** por el **comportamiento** de los **anteriores**.
- **Boosting** puede dañar performance en **datasets** ruidosos.
- El boosting se entrena hasta que consigue **clasificar bien** el máximo de ejemplos posibles.
- Boosting se lleva a cabo con adaboost y **escoge un valor** de confianza **αt**.
- Boosting concede **menor** peso a las **muestras** que están clasificados **correctamente** y el más **alto** a los **mal clasificados**.
- En boosting los **votos** son ponderados **unos mejores que otros**.
- En boosting los ejemplos más **cercanos** a la **frontera** de decisión reciben los pesos más **altos**.
- **Boosting** realiza un **muestreo** ponderado mientras que **Bagging no**.
- Los **votos ponderados** en boosting se realizan con clasificadores **débiles**.
- Boosting se usa a día de hoy en las cámaras (**detención de rostro**).
- Boosting es la técnica para **entrenar** varios **clasificadores débiles** para encontrar un clasificador mejor.

- **Bagging** ayuda a mejorar Redes neuronales y árboles de decisión.
- En Bagging los elementos del conjunto de entrenamiento clasificados por h(t) se pueden usar en h(t+1) y la hipótesis final no se selecciona el clasificador que mejor haya evaluado el conjunto de entrenamiento.

- **Diferencias entre Boosting y Bagging:** el **Boosting asigna pesos** a **cada registro** de entrenamiento y **Bagging** elige aleatoriamente los registros para **formar** los **subconjuntos**.
- Tanto boosting como bagging **mejoran clasificadores inestables** como por ejemplo las redes neuronales.
- **Boosting** pondera y da **más peso** a los ejemplos que **más cuestan** de **clasificar** y **bagging entrena un clasificador débil** con el subconjunto cogido T veces.

- Los modelos o clasificadores tienen **mismos pesos** en la formación de la **hipótesis final**.
- En el muestreo ponderado los ejemplos más **cercanos** a la **frontera** de **decisión** son los más **difíciles** de clasificar y recibirán los **pesos** más **altos**.

Adaboost.

- Es falso que "**i**" indexa **clasificadores** (**débiles**), mientras que "**t**" indexa **ejemplos**.
- Es un algoritmo utilizado para construir **clasificadores sólidos** utilizando la combinación lineal de clasificadores simples.
- Persigue **mantener** un peso en cada uno de los ejemplos de entrenamiento.
- Si obtenemos $\epsilon t = 0$ tenemos una **frontera perfecta**.
- Es un algoritmo adaptativo porque los subsecuentes **clasificadores** son **construidos** y **mejorados** en favor de los clasificadores **anteriores mal clasificados**.
- Pasos necesarios: **Calcular** error del **modelo** en el **set** de **entrenamiento** y **ponderar** todos los **sets** de entrenamiento de **igual** forma.
- **Boosting y Adaboost** están basados en **aprendizaje múltiple**.
- Ponderar el conjunto de clasificadores inicial de forma que en su conjunto podamos clasificar de forma correcta los ejemplos de nuestro entrenamiento.

Construyendo y usando Dt.

- El valor de confianza depende del error que se cometió en la **clasificación débil**.
- En Adaboost entrenamos un clasificador débil usando Dt para obtener ht.
- Cuando se actualiza la distribución D inicialmente, **cuando T>1** todos los **ejemplos** son **igualmente probables**.
- Usando Dt, **εt** es el **error** asociado a **ht**.
- El valor de αt surge de intentar optimizar el error asociado a ht, εt, y es ($\alpha t = 1/2 \ln (1 - \epsilon t) / \epsilon t$)
- **Primero entrenar un clasificador débil usando Dt** y obtener ht, escoger un valor de confianza αt y por último actualizar la distribución D.

TEMA 11. Percepción Visual Artificial.

Definiciones.

El modelo RGB supone **un problema** en el campo de sistemas inteligentes porque existen colores muy similares a simple vista que tienen una **representación RGB** muy **diferente** (alejada), lo cual puede suponer **un alto coste** computacional.

Cuando se **reduce la resolución** de una imagen, al perder píxeles promedia entre los **píxeles eliminados**.
En cuanto a los modelos de color **el tipo de imagen** que podemos obtener es una imagen **binaria**, valores 0 y 1.

Un método de **procesamiento robusto ante el ruido** de una imagen es bueno cuando genera los **mismos resultados** con o sin ruido. (**Google night mode 😉)**

El **error** que puede **degradar** la **imagen** suelen ser el **ruido**.
En **ampliación** de una imagen se **repiten píxeles**.

FALSO: Si ampliamos o reducimos una imagen el número de píxeles sigue siendo igual.

- **Convolución:**

El **nuevo** valor del **píxel** es el resultado de **sumar** la **multiplicación** de los **valores** de la máscara por los valores de los píxeles.

En el **proceso** de **captura** de una imagen mediante un **sensor** se produce una **discretización**.

Histograma.

- En histogramas podemos saber en una imagen qué número de **píxeles** comparten el **mismo valor**.
- Es **incorrecto** que la **ecualización** del histograma **consiste** en **comprimirlo** para mejorar el contraste.
- Si **aumentamos** el **valor** de todas las posiciones de la matriz de una imagen **aumentará** el **brillo** de la imagen.
- En una imagen con **poco contraste** podemos "**expandir**" el histograma.
- Si a una imagen se le **baja contraste y** se le **sube** el **brillo**, el histograma de la imagen **será más comprimido** y desplazado hacia la **derecha**.
- Para **aumentar** o **reducir** el **brillo** en una imagen tenemos que **aumentar** o **reducir** el **valor** de cada **píxel**.

La ecuación utilizada es:

- **Paso 1:** Se calcula el histograma de la imagen **h(k)**.

- **Paso 2:** Se calcula el histograma acumulado $Ha(k) = \sum(desde\ j = 0\ hasta\ k)\ h(j)$.

- **Paso 3:** se produce un mapeo con la siguiente fórmula $l'(x,y) = (Ha(l(x,y)) - \boldsymbol{minv}) * (L - 1)/(totalpix - \boldsymbol{minv})$

El proceso de binarización:

- Consiste en poner a **1** los píxeles que estén por **encima** de un **umbral** o entre dos umbrales y el **resto** a **0**.

Aristas (Edges).

- Son puntos de **alta derivada** en valor absoluto.
- Los tipos de **detectores** para **puntos** de **esquina** o **corners** están basados en "**edges**" o en "niveles de gris".
- En los filtros sobre ímagenes hay que **tener** en **cuenta el tamaño de la máscara** porque **puede tener** un **efecto no deseado** en el **resultado**, aunque hay veces que es lo que se pretende.

Detector de Canny.

Criterios (formalización):

- Buena **detección**: minimizar el número de falsos positivos y falsos negativos.
- Buena **localización**.
- **Respuesta única**.

Supuestos:

Ruido **gaussiano** y aristas tipo "**escalón**".

Resumen:

- En el **último paso** del trabajo el **algoritmo** realiza la unión de los píxeles, incorporando "**candidatos débiles**" que están 8-conectados a los píxeles "**probables**".
- El **filtro** que se usa para **suavizar** una imagen es el **Filtro Gaussiano**.
- El **valor** de **máximo** de la máscara de convolución aparece en el píxel central y disminuye hacia los extremos.
- Es el más efectivo a la hora de detectar **bordes** debido a su **eficacia**.
- Utilizando **trade-off** si **aumentamos sigma reducimos el ruido**, pero difuminamos los bordes y perdemos calidad en la localización.

Detector Nitzberg-Harris.

Para la construcción del detector de **Nitzberg-Harris** alguno de sus pasos son: **Calcular gradientes horizontal y vertical** para cada uno de los píxeles de la vecindad.

La **matriz A(x, y)** captura la estructura de intensidad de la vecindad local.

TEMA 12. Percepción automática.

En percepción automática en los modelos de color **el RGB presenta un problema importante** a la hora de segmentar por color, dos colores similares pueden aparecer lejos en el espacio de representación del modelo.

Transformada de Hough.

Motivación:

- Desarrollar técnicas que permitan **identificar primitivas** geométricas **sencillas** en una imagen.

Mecanismo:

- **Espacio paramétrico**. Los valores de los parámetros de la ecuación paramétrica **definen unívocamente** a **cada primitiva**. Por cada parámetro tenemos una dimensión en el espacio paramétrico y cada dimensión se "discretiza" en celdas.
- **Cada punto** de la imagen participa en un **proceso** de **votación**.

Análisis:

- **Ajuste de sensibilidad**: aunque es un factor de menor importancia, el ajusta del tamaño de celda y del número mínimo de votos es clave para evitar un elevado número de falsos positivos.

- Los puntos críticos son: **ineficiencia**, **ajuste de sensibilidad** y **solamente aplicable** a **primitivas sencillas**.

- La complejidad **espacial** y **temporal** de la ineficiencia son: O(N!) y O(N-1)! Respectivamente.

- En los puntos críticos el **método exhaustivo** solamente es aplicable a primitivas sencillas, haciéndose necesaria una discretización ligera para satisfacer los requerimientos de memoria.

FALSO: Emplear la transformada de Hough para determinar el reconocimiento de patrones geométricos inherentes a la imagen es suficiente para resolver problemas de iluminación.

Características SIFT.

SIFT trabaja de forma correcta sobre imágenes con **diferente iluminación**, ángulo de captura, etc.

Devuelve las **características** encontradas en la **imagen**.

Su orden es:

- Localización del punto. Escala. Orientación. Cálculo del descriptor. ->
  Encontrar la posición de los puntos, calcular el descriptor.

Tiene dos pasos:

- **Buscar** los puntos característicos de una imagen y **calcular** sus descriptores.

FALSO: encontrar para cada punto el centroide más cercano.

En la localización:

- La **multiescala** se consigue con una **diferencia gaussiana** (DoG).
- Los **puntos relevantes** son: **máximos** y **mínimos**.

Segmentación de imágenes.

Es el proceso de extraer zonas de la imagen con el **mismo color/nivel** de **gris/textura** para identificarlas automáticamente.

Algoritmo de las K-medias.

- Es un método de agrupamiento heurístico con **número** de **clases conocido** (**K**).
- En su modo probabilístico, se puede aplicar a un **número reducido de distribuciones**.
- Termina cuando ya no hay cambios en las pertenencias o se han realizado el **máximo** de **iteraciones** fijadas como parámetro.
- **Asignar** a cada clúster el **centroide más cercano no** es un **paso** de K-medias.
- Las **medias distribuciones** (**clústeres**) se pueden encontrar en el algoritmo.
- Se **inicializa** asignando las **medias** de manera **aleatoria**.
- Es necesario conocer el número de distribuciones distintas existentes.
- Los **clústeres representan** las **medias de las distribuciones** a partir de las cuales podemos clasificar una imagen con un conocimiento previo.
- Se **buscan k puntos** (medias) y **es necesario indicar** explícitamente el valor de **k**.
- **No es un punto crítico**: Una **mala inicialización** puede llevar más tiempo.

FALSO: cuando inicializamos el método de las K-medias no podemos redistribuir las medias de manera uniforme.

FALSO: la transformada de Hough usa coordenadas polares para la identificación de primitivas geométricas sencillas de imagen. Por los que el algoritmo de las K-medias se puede usar también.

Segmentación basada en regiones

La **técnica** de **segmentación** basada en regiones en general trabaja mejor en **imágenes con ruido**.

En el **método** de **Crecimiento** de **regiones** podemos empezar por cualquier píxel de la imagen, pero es mejor lanzar varios puntos de partida "**semillas**".
El **objetivo** es encontrar **regiones** de la imagen **homogéneas** según algún criterio.

Conlleva una complejidad **mayor**, al tener que manejar **alguna estructura** de datos adicional.

Existen **dos maneras** de hacerlo:

Crecimiento de regiones: La manera en la que se empieza con regiones pequeñas y después se hacen crecer o se mezclan siguiendo un criterio de similaridad.
Se aplica un **detector** de **aristas**. Los puntos cuyo valor de magnitud de gradiente estén próximos a cero serán "**valles**", es decir, **puntos** dentro de **regiones**. Usaremos estos puntos como "**semillas**".

Partición de regiones: Empezamos con **regiones grandes** y las **vamos dividiendo**, usando un criterio de **homogeneidad**.