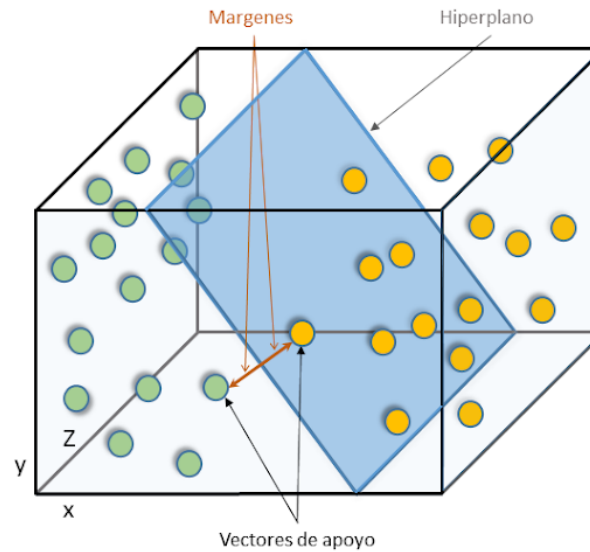




Universidad  
Rey Juan Carlos

Escuela Técnica Superior  
Ingeniería de Telecomunicación



# Visión Artificial

## 10. Reconocimiento de patrones

JOSÉ MIGUEL GUERRERO HERNÁNDEZ

EMAIL: [JOSEMIGUEL.GUERRERO@URJC.ES](mailto:JOSEMIGUEL.GUERRERO@URJC.ES)

# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

## 3. Aprendizaje no supervisado

- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

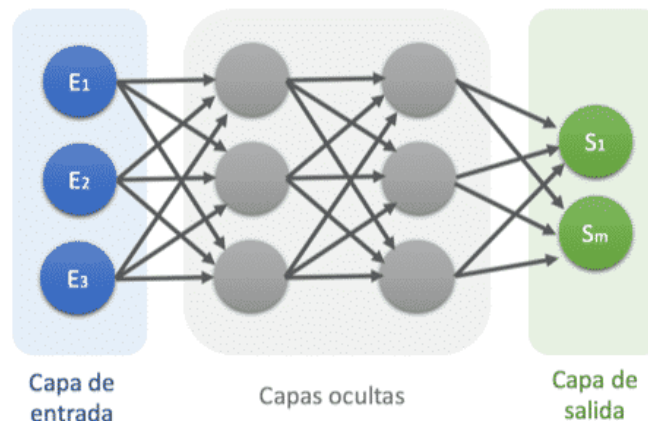
## 3. Aprendizaje no supervisado

- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

# 1. Introducción

- El **Machine Learning**, o aprendizaje automático o automatizado, tiene como objetivo desarrollar técnicas que permitan que los ordenadores, o máquinas con capacidad de procesamiento, aprendan
- Pero las máquinas no aprenden por sí mismas, están programadas para adaptar un algoritmo conforme reciben datos
  - Cuantos más datos reciben, mejores serán los algoritmos que crean, siempre y cuando los datos introducidos sean datos fiables y de calidad, ya que si se mete algún dato erróneo puede resultar en un mal aprendizaje
- Esto es posible en parte gracias a las redes neuronales



# 1. Introducción

---

- Mientras que con el **Machine Learning** hay que ayudar al ordenador indicándole qué son los datos introducidos, **aprendizaje supervisado**, en el **Deep Learning** se avanza en el concepto de **aprendizaje no supervisado**
- Los algoritmos DL son capaces de aprender sin intervención humana previa, sacando ellos mismos las conclusiones de todos los datos
- También es capaz de dotar a los objetos de un significado y manejar conceptos abstractos, no solo cosas materiales (son capaces de entender lo que están viendo)
- Al tener como base al Machine Learning, las **redes de neuronas** automáticas siguen presentes, realizando las mismas funciones, pero con un número de neuronas y capas bastante mayor y más sofisticadas
- Para poder llevar a cabo el aprendizaje de estas complejas y numerosas redes neuronales es necesario una gran capacidad de procesamiento, GPU

# 1. Introducción

---

- En general, los algoritmos de aprendizaje se pueden dividir en dos:
  - **Aprendizaje supervisado:**
    - Los algoritmos de aprendizaje supervisado ajustan el modelo o función de clasificación mediante un **conjunto de datos cuya clase es conocida**, llamado **conjunto de entrenamiento**
    - Formalmente este conjunto se representa como  $\{(x_1, y_1), \dots, (x_N, y_N)\}$ , donde  $x_i$  es el  $i$ -ésimo vector del conjunto de entrenamiento mientras  $y_i$  es la clase a la que pertenece dicho vector
    - Una vez ajustado el modelo, la función de clasificación es capaz de asignar la clase  $y_k$  a la que pertenecen nuevos elementos  $x_k$ . El conjunto de elementos de clase desconocida se llama **conjunto de test**
  - **Aprendizaje no supervisado:**
    - Los algoritmos de aprendizaje no supervisado tratan de ajustar un modelo de clasificación a partir de un **conjunto de datos no etiquetados** (de los que no se conocen las clases a las que pertenecen)  $\{x_1, \dots, x_N\}$
    - Para realizar este ajuste, el algoritmo debe encontrar de forma autónoma las características, regularidades y correlaciones de los datos, de forma que pueda separar las categorías similares entre sí

# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

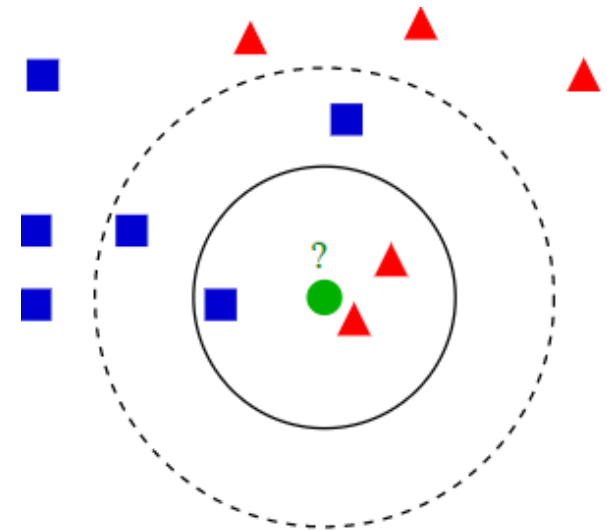
## 3. Aprendizaje no supervisado

- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

## 2. Aprendizaje supervisado: K-NN

- El algoritmo de los **k vecinos más cercanos** (**k-NN**, o **k Nearest Neighbour**) es un algoritmo de clasificación supervisado basado en criterios de vecindad
- En particular, k-NN se basa en la idea de que los nuevos ejemplos serán clasificados con la misma clase que tengan la mayor cantidad de vecinos más parecidos a ellos del conjunto de entrenamiento
- Este algoritmo sigue un procedimiento que seguimos cada uno de nosotros al ver un ejemplo nuevo: vemos a qué se parece más de lo que conocemos, y lo metemos en la misma bolsa





## 2. Aprendizaje supervisado: K-NN

---

- Su versión más **simple explora todo** el conocimiento almacenado en el conjunto de entrenamiento para determinar cuál será la clase a la que pertenece una nueva muestra, pero únicamente tiene en cuenta el vecino más próximo (más similar) a ella
- Es lógico pensar que es posible que no se esté aprovechando de forma eficiente toda la información que se podría extraer del conjunto de entrenamiento
- Con el objetivo de resolver esta posible deficiencia surge la **generalización de los  $k$  vecinos más cercanos (k-NN)**, en la que se utiliza la información suministrada por los  $k$  ejemplos del conjunto de entrenamiento más cercanos al que queremos clasificar

## 2. Aprendizaje supervisado: K-NN

- Una variante de este algoritmo consiste en ponderar la contribución de cada vecino de acuerdo a la distancia entre él y la muestra a ser clasificada, dando mayor peso a los vecinos más cercanos frente a los que puedan estar más alejados
- Si  $x$  es el ejemplo que queremos clasificar,  $V$  son las posibles clases de clasificación, y  $\{x_1, \dots, x_k\}$  es el conjunto de los  $k$  ejemplos de entrenamiento más cercanos, definimos el peso de  $x_i$  respecto a  $x$  como:

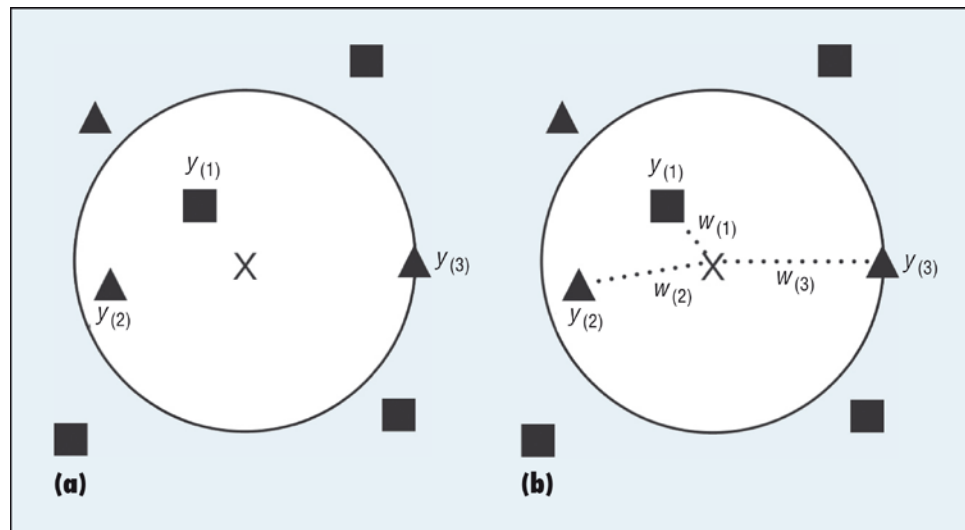
$$w_i = \frac{1}{d(x, x_i)^2}$$

- Y entonces la clase asignada a  $x$  es aquella que verifique que la suma de los pesos de sus representantes sea máxima, es decir:

$$\operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i$$

## 2. Aprendizaje supervisado: K-NN

- Esta variante es muy efectiva en muchos problemas prácticos. Es robusto ante el ruido de los datos y suficientemente efectivo en conjuntos de datos grandes



<https://youtu.be/gdS0V35GqgQ>

## 2. Aprendizaje supervisado: K-NN

---

- En el algoritmo k-NN existe el problema de que requiere de mucha memoria y tiempo de ejecución porque hay que almacenar continuamente todos los datos que definen el espacio de ejemplos inicial
- Sin embargo, es muy probable que muchas de las muestras iniciales no sean necesarias para clasificar las demás, ya que su información es redundante con las otras existentes
- Para mitigar esto:
  - **k-NN Condensado:** dado un orden en los datos de entrada, cada ejemplo del conjunto se clasifica por medio de k-NN haciendo uso únicamente de los datos anteriores; si la clasificación obtenida coincide con la real, ese ejemplo se elimina de los datos, si no, permanece. Observa que depende del orden dado a los datos y, además, tiene el problema de conservar los datos que introducen ruido al sistema
  - **k-NN Reducido:** es similar a la anterior, pero se comienza con el conjunto completo de datos, y se eliminan aquellos que no afectan a la clasificación del resto de datos de entrada. Al revés de lo que ocurre con la condensación, este método es capaz de eliminar las muestras que producen ruido, y guarda aquellas que son críticas para la clasificación

# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

## 3. Aprendizaje no supervisado

- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

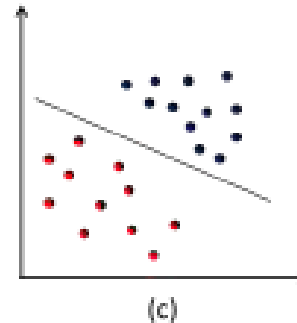
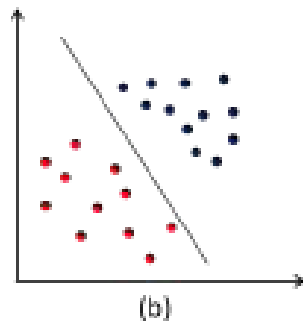
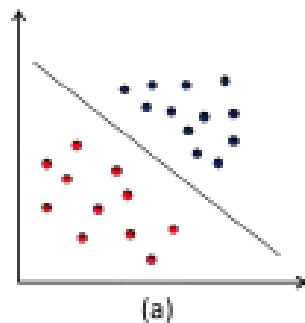
## 2. Aprendizaje supervisado: SVM

---

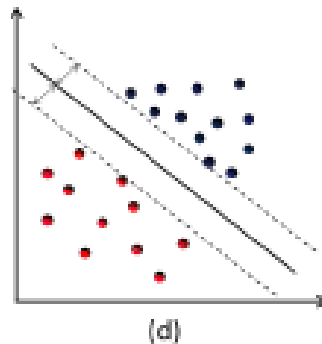
- Las **Máquinas de Vectores Soporte** (SVM por sus siglas en inglés Support Vector Machines) (Vapnik, 1995) constituyen una de las técnicas de clasificación supervisada más potentes
- Se trata de un **clasificador biclase** (esto es, permite clasificar en dos clases), aunque la mayoría de implementaciones del mismo (LIBSVM, LIBLINEAR) soportan clasificación multi-clase
- Al igual que cualquier clasificador supervisado biclase, el objetivo de un SVM es **inferir una frontera de decisión** (lineal o no) en el espacio de características, de modo que las observaciones posteriores se clasifiquen automáticamente en uno de los dos grupos definidos por dicha frontera (también conocida como **hiperplano**)

## 2. Aprendizaje supervisado: SVM

- La particularidad de SVM es que trata de generar dicho hiperplano de modo que maximice su separación con cada uno de los grupos



**Diferentes hiperplanos**



**Hiperplano de margen máximo**

## 2. Aprendizaje supervisado: SVM

- La forma general de la función de decisión SVM viene dada por la expresión:

$$f(x) = \sum_{k=1}^n \alpha_k y_k H(x_k, x) - b$$

donde  $x$  es un vector tridimensional genérico cuyas componentes pueden ser los tres valores espectrales  $R, G, B$  de la imagen original y  $b$  es un sesgo cuyo valor constante se establece más adelante

- Los parámetros  $\alpha_k$  con  $k = 1, \dots, n$  son la solución para el siguiente problema de optimización cuadrática:

$$Q(\alpha) = \alpha_k - \frac{1}{2} \sum_{k,h=1}^n \alpha_k \alpha_h y_k y_h H(x_k, x_h)$$



## 2. Aprendizaje supervisado: SVM

- Bajo las siguientes restricciones:

$$\sum_{k=1}^n \alpha_k y_k = 0 \quad 0 \leq \alpha_k \leq \frac{C}{2} \quad k = 1, \dots, n$$

donde  $H$  es el producto escalar del núcleo mencionado anteriormente, y  $C$  es un parámetro de normalización

- La distancia mínima de los hiperplanos de separación entre las clases a los puntos de datos se conoce como margen  $\tau$
- Un hiperplano de separación es óptimo si el margen es máximo
- La distancia entre un plano de separación y un patrón  $x_k$  dado, es  $y_k = |f(x_k)| / \|w\|$ , donde  $w$  es un vector perpendicular al hiperplano definido por

$$w = \sum_{k=1}^n \alpha_k y_k x_k$$

## 2. Aprendizaje supervisado: SVM

---

- El problema de encontrar el hiperplano óptimo es el de encontrar aquella  $w$  que maximiza  $\tau$
- Existen infinitas soluciones que difieren sólo en la escala de  $w$ , de modo que para limitar las soluciones se fija la escala en el producto de  $\tau$  y el vector normal de  $w$ :

$$\tau = \frac{1}{\|w\|}$$

donde  $\|\cdot\|$  es una norma, siendo elegida la norma euclídea por lo general

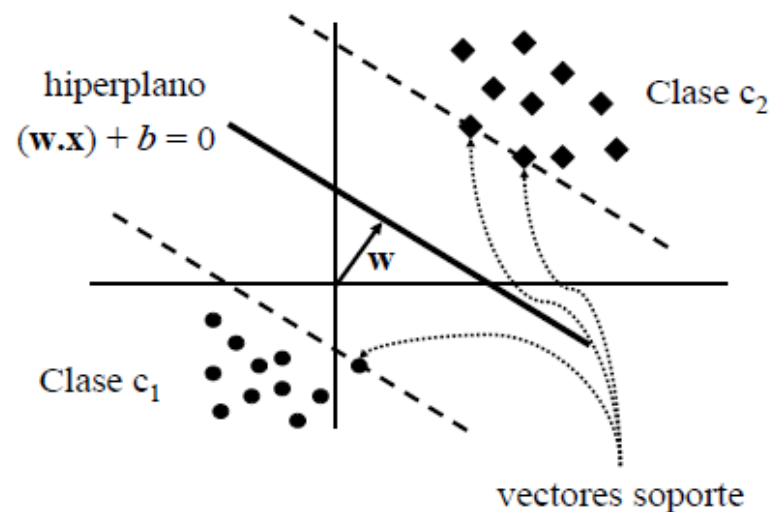
## 2. Aprendizaje supervisado: SVM

- El parámetro de sesgo  $b$  mencionado anteriormente se calcula como:

$$b = -\frac{1}{s} \sum_{k=1}^s (ws_k - y_k)$$

donde  $s_k$  son los  $s$  vectores soporte, es decir, los patrones de datos  $s$  asociados a  $\alpha_k$  distintos de cero, e  $y_k$  se inicializa con respecto al patrón que le corresponde en cada clase  $\{+1, -1\}$

- La salida del sistema SVM es  $\{+1, -1\}$ , donde  $+1$  y  $-1$  están asociados a las clases  $C_1$  y  $C_2$  respectivamente



# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

## 3. Aprendizaje no supervisado

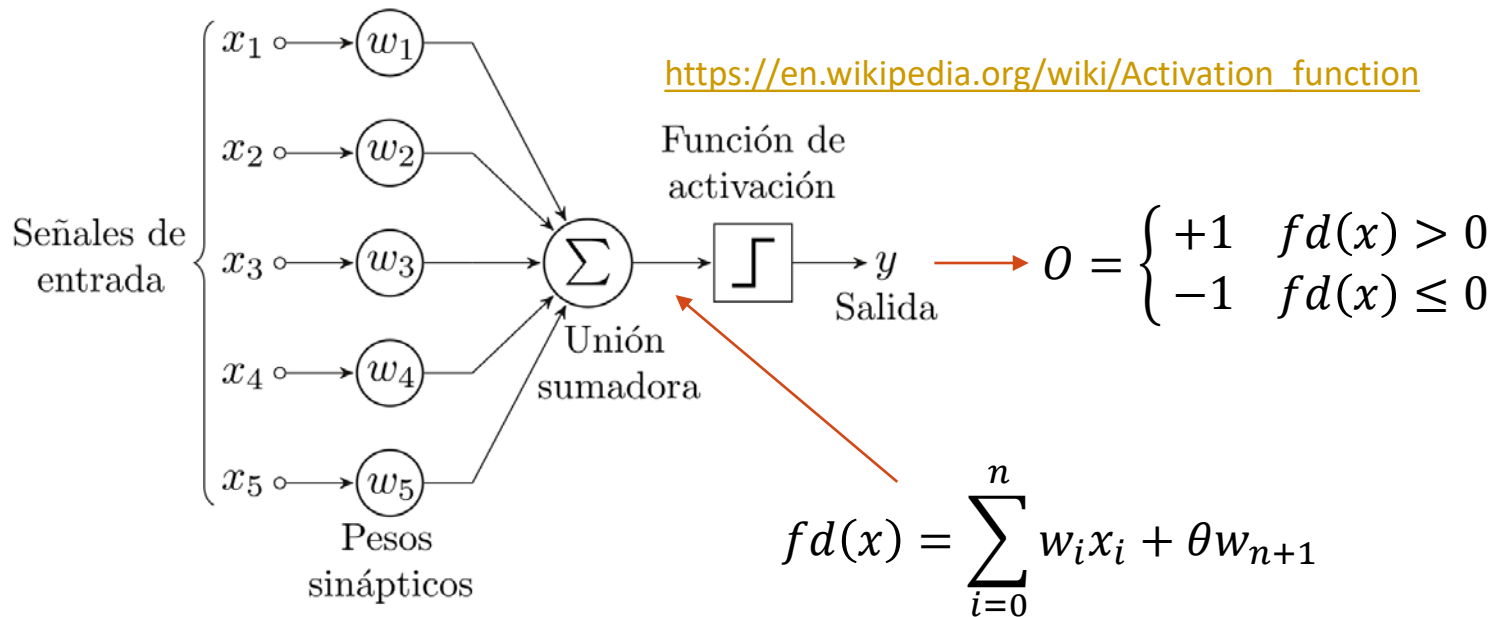
- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

## 2. Aprendizaje supervisado: Perceptrón

- **Red neuronal monocapa: Perceptrón simple**

- La red neuronal monocapa se corresponde con la red neuronal más simple, está compuesta por una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan los diferentes cálculos



## 2. Aprendizaje supervisado: Perceptrón

---

- **Red neuronal monocapa: Perceptrón simple**

- El **funcionamiento** del perceptrón es muy **sencillo**, simplemente lee los valores de entrada, suma todos las entradas de acuerdo a unos pesos y el resultado lo introduce en una función de activación que genera el resultado final
- El entrenamiento del perceptrón no es más que **determinar los pesos sinápticos y el umbral** que mejor hagan que la entrada se ajuste a la salida
- Para la determinación de estas variables, se sigue un **proceso adaptativo**. El proceso comienza con valores aleatorios y se van modificando estos valores según la diferencia entre los valores deseados y los calculados por la red
- En resumen, el perceptrón aprende de manera **iterativa**
- Hay que tener en cuenta que el **perceptrón solo es capaz de representar funciones lineales** debido a que no dispone de capas ocultas como por ejemplo el perceptrón multicapa

## 2. Aprendizaje supervisado: Perceptrón

- **Red neuronal monocapa: Perceptrón simple**

- **Entrenamiento:**

1. **Inicialización de los pesos y del umbral:** inicialmente se asignan valores aleatorios a cada uno de los pesos  $w_i$   $i = 1, 2, \dots, n, n + 1$  y al bias  $\theta$
2. **Presentación de un nuevo par** (Entrada, Salida esperada): presentar un nuevo patrón de entrada  $x_p = \{x_1, x_2, \dots, x_n\}$  junto con la salida esperada  $fd_i(k)$
3. **Cálculo de la salida actual:**  $y(k) = f[w^t x_i - w_{n+1}]$ , siendo en este caso  $f$  la función de transferencia escalón
4. **Adaptación de los pesos:**  $w_i(k + 1) = w_i(k) + \alpha(k)[fd_i(k) - y(k)]x_i$  La salida esperada  $fd_i(k)$  es 1 si el patrón pertenece a la clase  $A$  y  $-1$  si es de la clase  $B$ . La tasa de aprendizaje viene representada por  $\alpha(k)$
5. **Criterio parada:** si no hay convergencia (los pesos cambian), volver al paso 2

- **Decisión:**

$$x \in c_i \quad \text{si} \quad fd_i(x) > fd_j(x) \quad \forall j = 1, 2 \quad j \neq i$$

## 2. Aprendizaje supervisado: Perceptrón

- Red neuronal monocapa: Perceptrón simple

- Ejemplo

- Entrenamiento:

	$c_1 \rightarrow +1$		
$R$	200	210	215
$G$	160	170	172
$B$	120	130	133

	$c_2 \rightarrow -1$		
	90	92	87
	130	138	128
	60	54	66

convergencia

$$\alpha = 10^{-4}$$

$$\|w(k+1) - w(k)\| < \varepsilon$$

$$\varepsilon = 10^{-4}$$

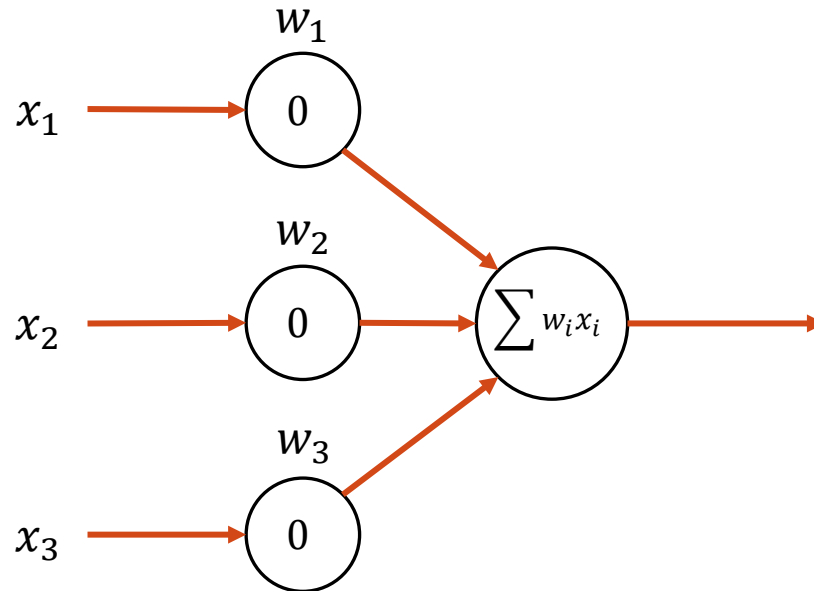
- Clasificación:

$$A \equiv (208, 170, 135)$$



## 2. Aprendizaje supervisado: Perceptrón

- Red neuronal monocapa: Perceptrón simple
  - Entrenamiento:
  - Inicialización:  $w^t(1) = (0,0,0)$ ;  $\alpha = 0.0001$



## 2. Aprendizaje supervisado: Perceptrón

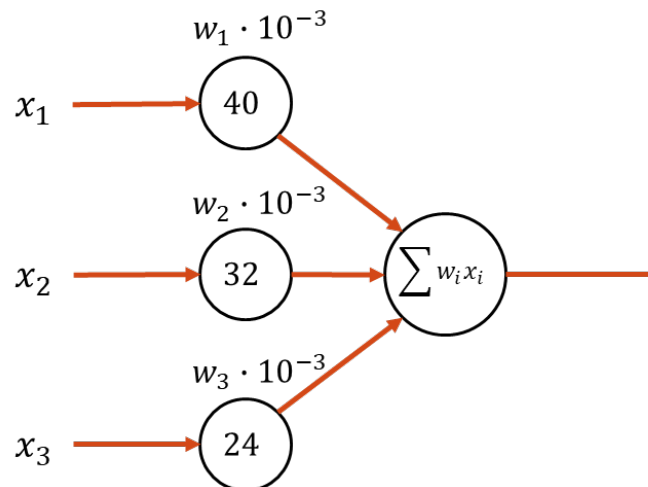
- Red neuronal monocapa: Perceptrón simple

- Entrenamiento:

- Inicialización:  $w^t(1) = (0,0,0)$ ;  $\alpha = 0.0001$

- Iteración 1:

- Patrón  $x = \{200,160,120\}$ :  $w^t x = 0$ ;  $O = -1$ ,  $error = (fd_i - O) = 1 + 1 = 2$  si se modifican los pesos:  $w^t(2) = w^t(1) + \alpha(fd_i - O)x = 10^{-3}(40,32,24)$



## 2. Aprendizaje supervisado: Perceptrón

- Red neuronal monocapa: Perceptrón simple

- Entrenamiento:

- Inicialización:  $w^t(1) = (0,0,0)$ ;  $\alpha = 0.0001$

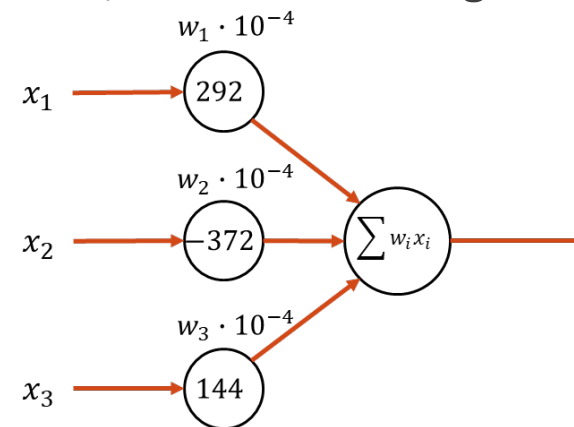
- Iteración 1:

1. Patrón  $x = \{200,160,120\}$ :  $w^t x = 0$ ;  $O = -1$ ,  $error = (fd_i - O) = 1 + 1 = 2$  si se modifican los pesos:  $w^t(2) = w^t(1) + \alpha(fd_i - O)x = 10^{-3}(40,32,24)$
2. Patrón  $x = \{210,170,130\}$ :  $w^t x = 17$ ;  $O = 1$ ,  $error = (fd_i - O) = 1 - 1 = 0$  no se modifican los pesos:  $w^t(3) = w^t(2)$
3. Patrón  $x = \{215,172,133\}$ :  $w^t x = 17$ ;  $O = 1$ ,  $error = (fd_i - O) = 1 - 1 = 0$  no se modifican los pesos:  $w^t(4) = w^t(3)$
4. Patrón  $x = \{90,130,60\}$ :  $w^t x = 9.2$ ;  $O = 1$ ,  $error = (fd_i - O) = -1 - 1 = -2$  si se modifican los pesos:  $w^t(5) = w^t(4) + \alpha(fd_i - O)x = 10^{-3}(22,6,12)$
5. Patrón  $x = \{92,138,54\}$ :  $w^t x = 3.5$ ;  $O = 1$ ,  $error = (fd_i - O) = -1 - 1 = -2$  si se modifican los pesos:  $w^t(6) = w^t(5) + \alpha(fd_i - O)x = 10^{-3}(4, -22, 1)$
6. Patrón  $x = \{87,128,66\}$ :  $w^t x = -2.4$ ;  $O = -1$ ,  $error = (fd_i - O) = -1 + 1 = 0$  no se modifican los pesos:  $w^t(7) = w^t(6)$

## 2. Aprendizaje supervisado: Perceptrón

- Red neuronal monocapa: Perceptrón simple
  - Entrenamiento:
  - El proceso converge en la iteración 4, resultando el siguiente vector de pesos:

$$w = 10^{-4}(292, -372, 144)$$



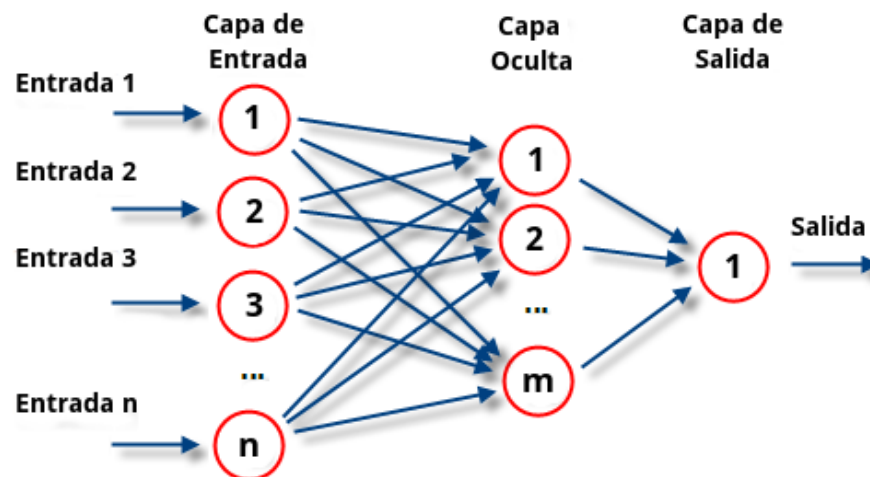
- Clasificación:
- Con dicho vector de pesos se clasifican los vectores dados:

$$w^t A = 10^{-4}(292, -372, 144) \begin{pmatrix} 208 \\ 170 \\ 135 \end{pmatrix} = 1.6936 > 0 \rightarrow A \text{ pertenece a la clase } c_1$$

## 2. Aprendizaje supervisado: Perceptrón

- **Red neuronal multicapa: Perceptrón multicapa**

- La red neuronal multicapa es una generalización de la red neuronal monocapa, la diferencia reside en que mientras la red neuronal monocapa está compuesta por una capa de neuronas de entrada y una capa de neuronas de salida, esta dispone de un conjunto de capas intermedias (capas ocultas) entre la capa de entrada y la de salida
- Dependiendo del número de conexiones que presente la red, ésta puede estar total o parcialmente conectada



## 2. Aprendizaje supervisado: Perceptrón

---

- **Red neuronal multicapa: Perceptrón multicapa**

- El perceptrón multicapa evoluciona el perceptrón simple y para ello incorpora capas de neuronas ocultas, con esto consigue representar **funciones no lineales**
- El perceptrón multicapa esta compuesto por una capa de entrada, una o varias capas de salida y  $n$  capas ocultas entremedias
- Se caracteriza por tener salidas disjuntas pero relacionadas entre sí, de tal manera que la salida de una neurona es la entrada de la siguiente
- En el perceptrón multicapa se pueden diferenciar 2 fases:
  - **Propagación:** en la que se calcula el resultado de salida de la red desde los valores de entrada hacia delante
  - **Aprendizaje:** en la que los errores obtenidos a la salida del perceptrón se van propagando hacia atrás (**backpropagation**) con el objetivo de modificar los pesos de las conexiones para que el valor estimado de la red se asemeje cada vez más al real, esta aproximación se realiza mediante la función gradiente del error

# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

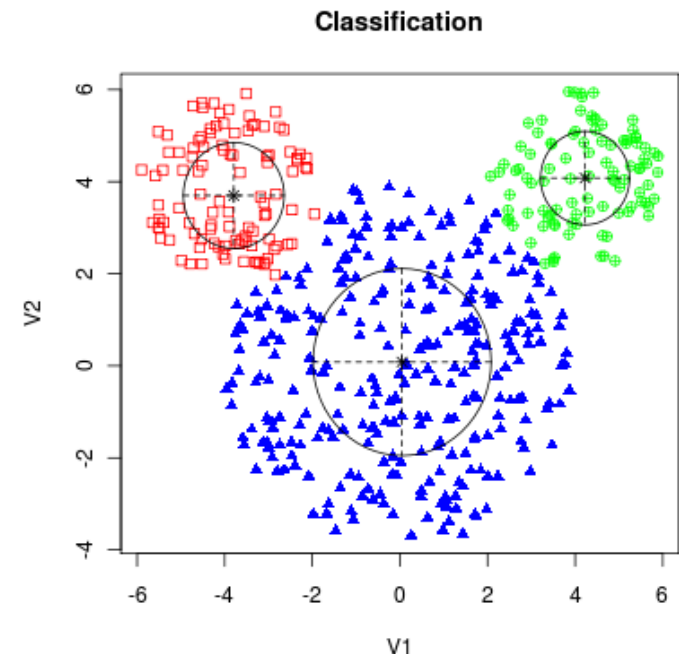
## 3. Aprendizaje no supervisado

- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

# 3. Aprendizaje no supervisado: K-means

- El algoritmo de  $K - means$  es aplicable en los casos en que tengamos una representación de nuestros datos como elementos en un espacio métrico
- El algoritmo de  $K - means$  intenta encontrar una partición de las muestras en  $K$  agrupaciones, de forma que cada ejemplo pertenezca a una de ellas, concretamente a aquella cuyo centroide esté más cerca
- El mejor valor de  $K$  para que la clasificación separe lo mejor posible los ejemplos no se conoce a priori, y depende completamente de los datos con los que trabajemos





### 3. Aprendizaje no supervisado: K-means

- En este caso, **no tenemos un conocimiento a priori** que nos indique cómo deben agruparse ninguno de los datos de que disponemos, es decir, **no hay un protocolo externo que nos indique lo bien o mal que vamos a realizar la tarea**, ningún criterio supervisa la bondad de nuestras soluciones
- Pero eso no significa que nosotros no podamos introducir una medida de bondad, aunque sea artificial y subjetiva
- En este caso, el algoritmo de las *K – means* va a intentar minimizar la varianza total del sistema, es decir, si  $c_i$  es el centroide de la agrupación  $i$ -ésima, y  $\{x_j^i\}$  es el conjunto de ejemplos clasificados en esa agrupación, entonces intentamos minimizar la función:

$$\sum_i \sum_j d(x_j^i, c_i)^2$$

## 3. Aprendizaje no supervisado: K-means

---

- Intuitivamente, cuanto más pequeño sea este valor, más agrupados están los ejemplos en esos conjuntos
- Pero observemos que el número de conjuntos no viene dado por el algoritmo, sino que hemos de decidirlo antes de ejecutarlo
- A pesar de que el problema se plantea como una optimización, existe un algoritmo muy sencillo que devuelve un resultado similar
- Fijado  $K$ , los pasos que sigue el algoritmo son los siguientes:
  1. Seleccionar al azar  $K$  puntos del conjunto de datos como centros iniciales de los grupos
  2. Asignar el resto de ejemplos al centro más cercano (tenemos  $K$  agrupaciones iniciales)
  3. Calcular el centroide de los grupos obtenidos
  4. Reasignar los centros a estos centroides
  5. Repetir desde el paso 2 hasta que no haya reasignación de centros (o los últimos desplazamientos estén por debajo de un umbral y no haya cambios en las agrupaciones obtenidas)

<https://youtu.be/74rv4snLI70>

# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

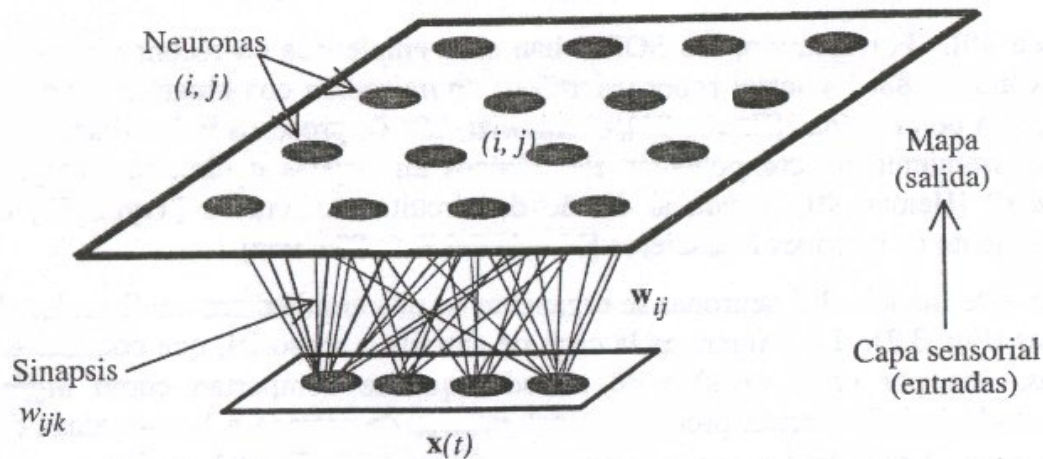
## 3. Aprendizaje no supervisado

- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

### 3. Aprendizaje no supervisado: SOM

- Un modelo SOM (Self-Organizing Maps) está compuesto por dos capas de neuronas
- La capa de entrada (formada por  $N$  neuronas, una por cada variable de entrada) se encarga de recibir y transmitir a la capa de salida la información procedente del exterior
- La capa de salida (formada por  $M$  neuronas) es la encargada de procesar la información y formar el mapa de rasgos



### 3. Aprendizaje no supervisado: SOM

---

- Las conexiones entre las dos capas que forman la red son siempre hacia delante, es decir, la información se propaga desde la capa de entrada hacia la capa de salida
- Cada neurona de entrada  $i$  está conectada con cada una de las neuronas de salida  $j$  mediante un peso  $w_{ij}$
- Las neuronas de salida tienen asociado un vector de pesos  $W_j$  llamado vector de referencia (o *codebook*), debido a que constituye el vector prototipo (o promedio) de la categoría representada por la neurona de salida  $j$
- Las neuronas adyacentes pertenecen a una vecindad  $N_j$  de la neurona  $j$

### 3. Aprendizaje no supervisado: SOM

---

- La topología y el número de neuronas permanece fijo desde el principio
- El número de neuronas determina la suavidad de la proyección, lo cual influye en el ajuste y capacidad de generalización del SOM
- Durante la fase de entrenamiento, el SOM forma una red elástica que se pliega dentro de la nube de datos originales. El algoritmo controla la red de modo que tiende a aproximar la densidad de los datos
- Los vectores de referencia del *codebook* se acercan a las áreas donde la densidad de datos es alta

### 3. Aprendizaje no supervisado: SOM

- Algoritmo de Kohonen:

- **Paso 1.** Un vector  $x$  es seleccionado al azar del conjunto de datos y se calcula su distancia (similitud) a los vectores de referencia, usando, por ejemplo, la distancia euclídea:

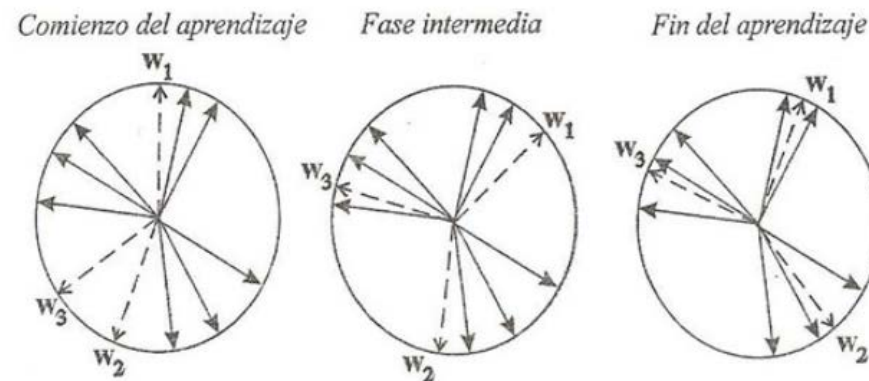
$$\|x - m_c\| = \min_j \{\|x - m_j\|\}$$

- **Paso 2.** Una vez que se ha encontrado el vector más próximo o BMU (Best Matching Unit) el resto de vectores de referencia se actualizan. El BMU y sus vecinos (en sentido topológico) se mueven cerca del vector  $x$  en el espacio de datos. La magnitud de dicha atracción está regida por la *tasa de aprendizaje*  $\alpha$
- Mientras se va produciendo el proceso de actualización y nuevos vectores se asignan al mapa, la tasa de aprendizaje decrece gradualmente hacia cero. Junto con ella también decrece el radio de vecindad también
- La regla de actualización para el vector de referencia dado  $i$  es la siguiente:

$$m_j(t+1) = \begin{cases} m_j(t) + \alpha(t) (x(t) - m_j(t)) & j \in N_c(t) \\ m_j(t) & j \notin N_c(t) \end{cases}$$

# 3. Aprendizaje no supervisado: SOM

- Algoritmo de Kohonen:
  - Los pasos 1 y 2 se van repitiendo hasta que el entrenamiento termina. El número de pasos de entrenamiento se debe fijar antes a priori, para calcular la tasa de convergencia de la función de vecindad y de la tasa de aprendizaje
  - Una vez terminado el entrenamiento, el mapa ha de ordenarse en sentido topológico:  $n$  vectores topológicamente próximos se aplican en  $n$  neuronas adyacentes o incluso en la misma neurona





### 3. Aprendizaje no supervisado: SOM

---

- Medidas de calidad del mapa y precisión del mapa:
  - Una vez que se ha entrenado el mapa, es importante saber si se ha adaptado adecuadamente a los datos de entrenamiento. Como medidas de calidad de los mapas se considera la precisión de la proyección y la preservación de la topología
  - La medida de precisión de la proyección describe cómo se adaptan o responden las neuronas a los datos. Habitualmente, el número de datos es mayor que el número de neuronas y el error de precisión es siempre diferente de 0
  - Para calcular la precisión de la proyección se usa el error medio de cuantificación sobre el conjunto completo de datos:

$$\varepsilon_q = \frac{1}{N} \sum_{i=1}^N \|x - m_c\|$$

### 3. Aprendizaje no supervisado: SOM

---

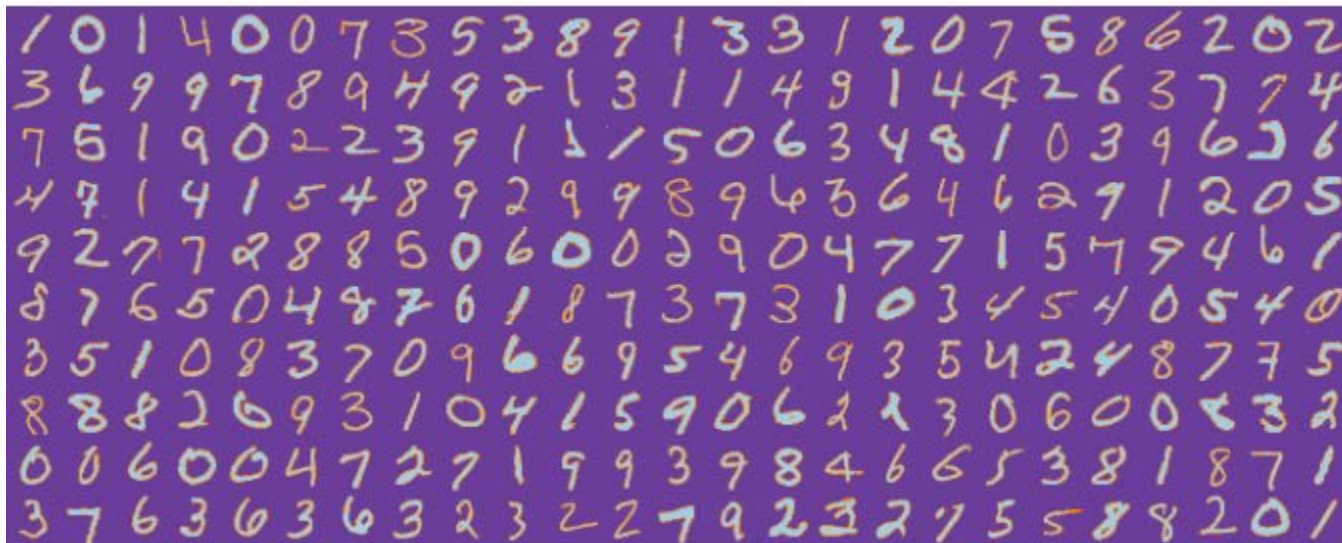
- Medidas de calidad del mapa y precisión del mapa:
  - La medida de preservación de la topología describe la manera en la que el SOM preserva la topología del conjunto de datos. Esta medida considera la estructura del mapa. En un mapa que esté *retorcido* de manera extraña, el error topográfico es grande incluso si el error de precisión es pequeño
  - Una manera simple de calcular el error topográfico es:

$$\varepsilon_q = \frac{1}{N} \sum_{i=1}^N u(x_k)$$

donde  $u(x_k)$  es igual a 1 si el primer y segundo BMUs de  $x_k$  no están próximos el uno al otro. De otro modo,  $u(x_k)$  es igual a 0

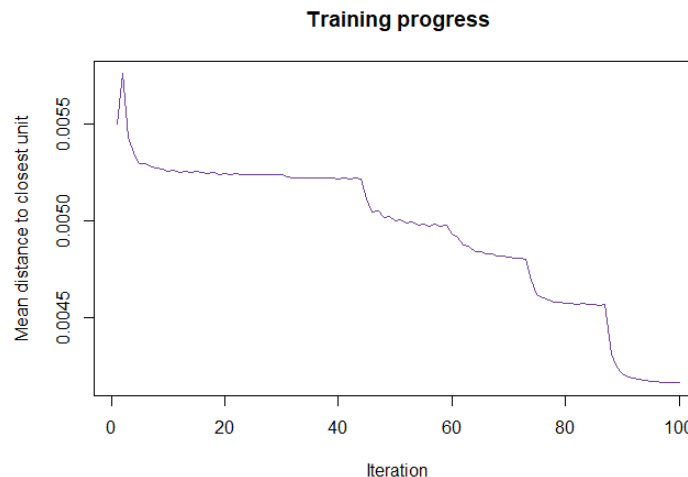
### 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - Aprendizaje de números escritos a mano
  - 10 clases posibles, dígitos del cero al nueve (imágenes 28x28)
  - El rango de las imágenes tenemos que varia entre 0 y 255, se normaliza



# 3. Aprendizaje no supervisado: SOM

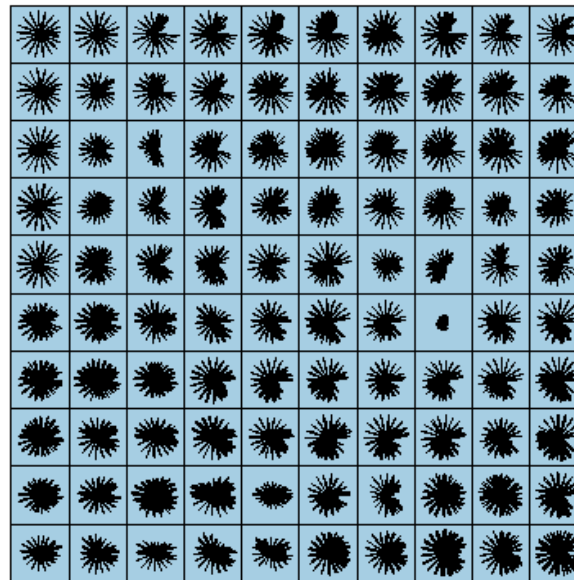
- Ejemplo (datos del MNIST):
  - Entrenamiento con 29404 muestras
  - Número de iteraciones 100
  - Tasa de aprendizaje  $\alpha$  comienza en 0.7 y decrece hasta 0.01
  - Vecindad circular de radio 7
  - Capa de salida formada por 10x10 elementos



# 3. Aprendizaje no supervisado: SOM

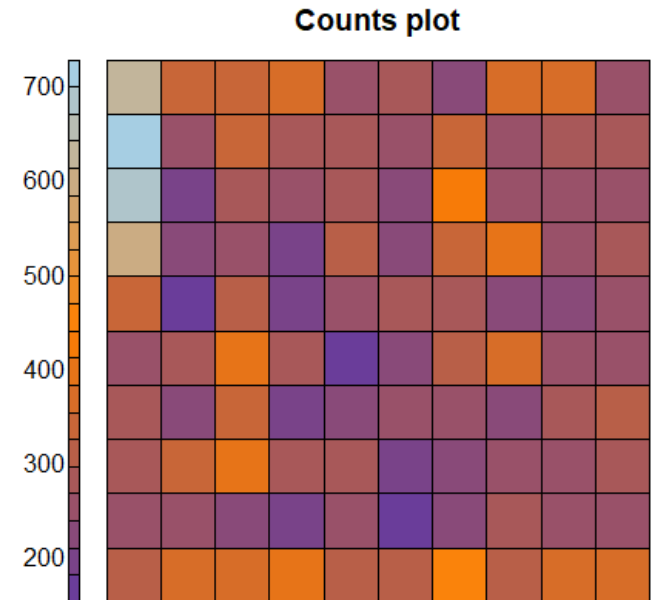
- Ejemplo (datos del MNIST):
  - *codebook* del modelo: refleja como influye cada uno de los 784 píxeles a cada una de las neuronas de salida. Se observa que neuronas cercanas tienden a tener distribuciones similares

Codes plot



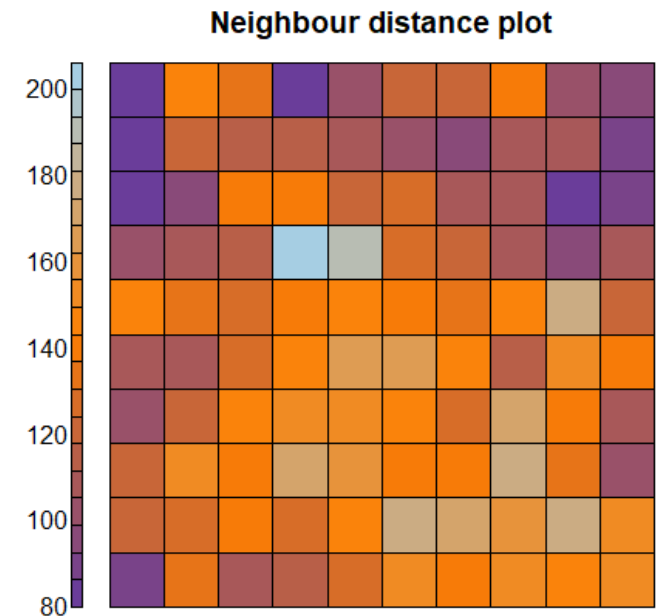
### 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - Conteo de observaciones mapeadas a cada neurona, esto influye en la capacidad para distinguir entre distintos tipos de observaciones:
    - Si hay sectores con muchos conteos de observaciones quiere decir que un gran número de observaciones presentan las mismas características, esto podría ser negativo en su caso extremo, ya que si un sector detecta la mayoría de las observaciones no entregaría información útil para diferenciar clases
    - Lo mismo ocurre para neuronas sin observaciones que no entregarían información
  - Este no es el caso, aunque hay valores máximos, se observan variaciones de color en la figura, lo que representa que hay cierta variabilidad en los datos



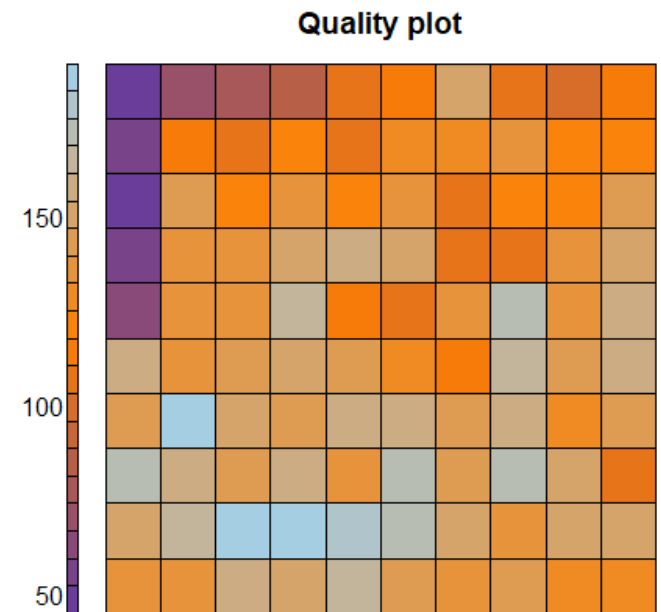
# 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - El gráfico de distancia entre neuronas vecinas es útil para visualizar posibles fronteras entre zonas y tener una idea de donde se agruparían distintos grupos. Estos grupos son calculados durante el entrenamiento y no tienen que ver con las clases definidas, ya que el entrenamiento es no supervisado
  - Donde exista un cambio brusco de tonalidad es posible que exista una frontera



# 3. Aprendizaje no supervisado: SOM

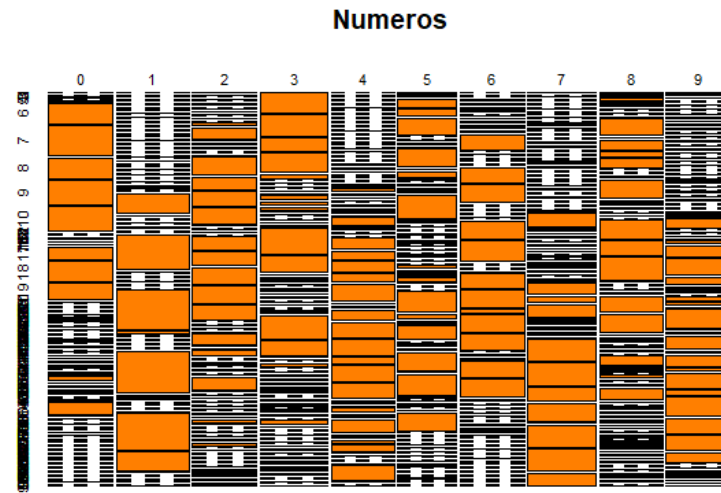
- Ejemplo (datos del MNIST):
  - La calidad presente en el modelo se puede representar utilizando la distancia de las observaciones al *codebook* final, mientras menor distancia mejor representación de las observaciones





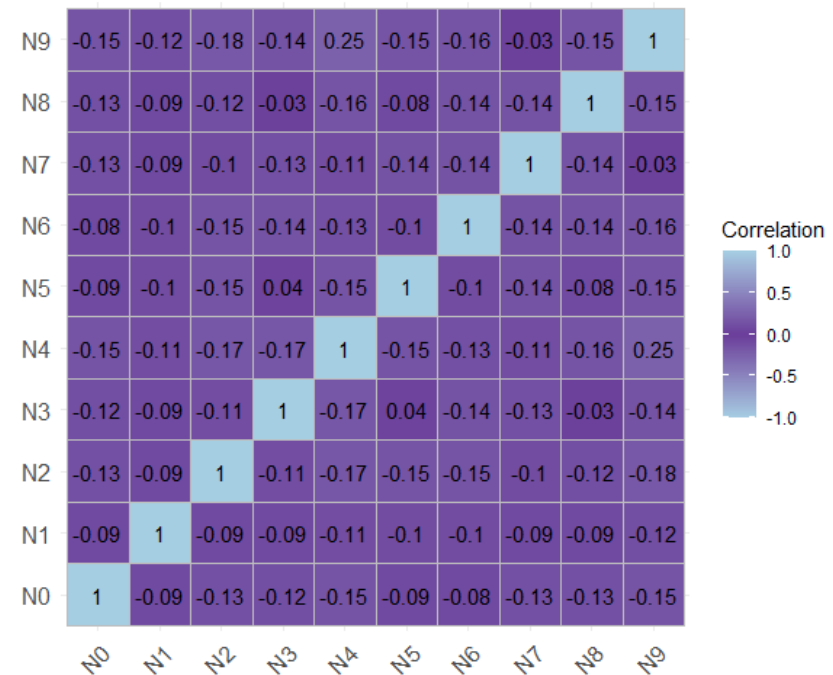
# 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - Para observar cómo quedan distribuidas las clases (cada número) en el mapa, se considera el porcentaje de ocurrencia de cada clase por neurona
  - Lo anterior se realiza, primero definiendo cual es la neurona del mapa con el mayor valor por observación, por ejemplo, para la primera observación la neurona con mayor valor es la 92, por lo que a esta neurona se le asigna el valor del target que en este caso es el numero 1, así para todas las observaciones



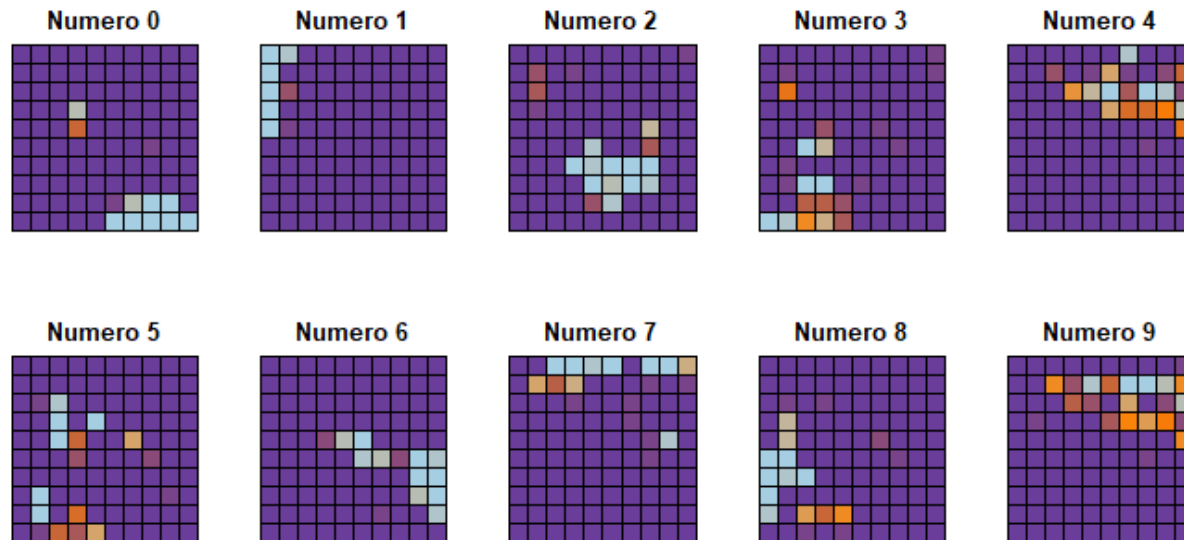
# 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - Se observa que cada clase activa neuronas específicas, ahora vamos a ver si estas distribuciones están relacionadas o no y por lo tanto si el mapa es útil para extraer características que diferencien cada clase
  - Al realizar la correlación entre los datos tenemos que los datos generados por el modelo para cada clase no están correlacionados, lo que es muy útil para clasificar
  - Los números con mayor correlación son el 4 y el 9 con una correlación 0.253



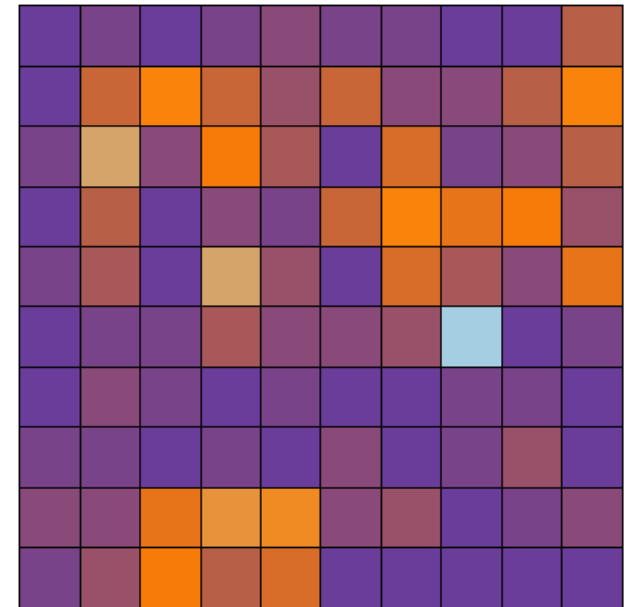
# 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - Ahora observando las zonas del mapa asociada a cada clase tenemos que cada clase se ubica en zonas relativamente diferentes del resto de las clases, esto indica que el modelo separa las características que diferencian a cada clase



### 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - La desviación estándar de cada neurona por clase presenta algunas zonas conflictivas
  - El color morado representa una desviación máxima y a medida que se acerca al celeste (pasando por naranja) se acerca a 0
  - Las neuronas con **menor desviación** por clase se muestran en tonos naranja y celeste, estas representan zonas difusas donde se ubican **características mas generales y pueden pertenecer a varias clases**, por lo que genera desviación menor entre clases
  - En este caso particular las zonas con baja desviación están asociadas por un lado una zona grande del 9 y el 4, donde hasta un ser humano podría tener problemas en diferenciar; por otro en una zona menor asociada al 3, 5 y 8, números que también podrían llegar a ser confundidos



### 3. Aprendizaje no supervisado: SOM

- Ejemplo (datos del MNIST):
  - Como se observa en la matriz de confusión, los números 4 y 9 presentan algunos problemas, así como el 3 y el 8 con el numero 5. Estos números corresponden a las zonas conflictivas de la anterior
  - De todas maneras la clasificación entrega un valor de precisión igual a 0.849 y un valor de kappa igual a 0.8322 lo que es bastante bueno para ser un modelo no supervisado

Confusion Matrix and Statistics

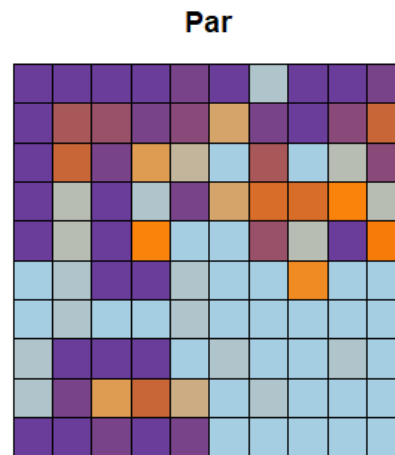
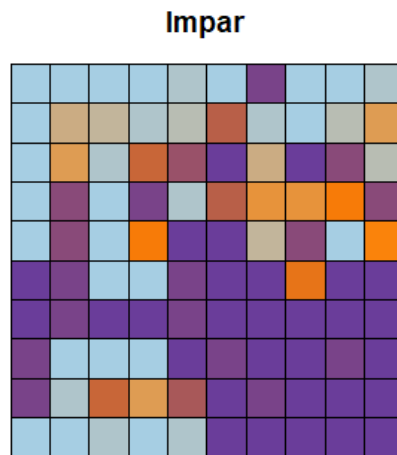
Prediction \ Reference	0	1	2	3	4	5	6	7	8	9
0	1165	0	12	20	0	30	30	2	3	6
1	0	1366	16	2	9	3	2	13	6	5
2	18	5	1117	21	23	36	31	31	18	14
3	4	15	23	1047	0	150	1	1	51	11
4	2	1	4	1	867	8	1	26	12	174
5	20	2	1	82	4	859	5	3	92	12
6	25	2	11	4	20	20	1168	0	8	3
7	2	2	32	16	11	2	0	1130	7	43
8	3	6	37	95	1	16	3	10	999	12
9	0	6	0	17	286	14	0	104	22	976

Overall Statistics

Accuracy : 0.849  
 95% CI : (0.8426, 0.8552)  
 No Information Rate : 0.1115  
 P-Value [Acc > NIR] : < 2.2e-16  
  
 Kappa : 0.8322

### 3. Aprendizaje no supervisado: SOM

- Otros ejemplos:
- Números pares o impares



Confusion Matrix and Statistics

	Reference	
Prediction	0	1
0	5881	594
1	543	5578

Accuracy : 0.9097  
 95% CI : (0.9046, 0.9147)  
 No Information Rate : 0.51  
 P-Value [Acc > NIR] : <2e-16

Kappa : 0.8194

### 3. Aprendizaje no supervisado: SOM

- Otros ejemplos:
- Reconocimiento facial



# 3. Aprendizaje no supervisado: SOM

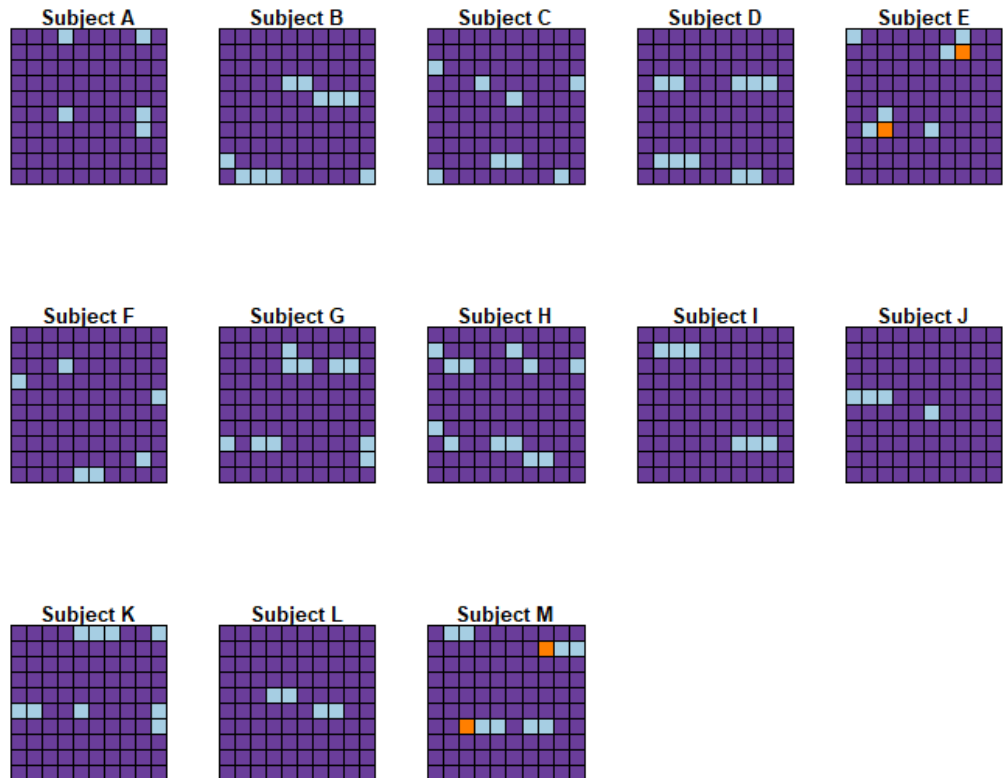
- Otros ejemplos:
- Reconocimiento facial

Confusion Matrix and Statistics

		Reference												
Prediction		A	B	C	D	E	F	G	H	I	J	K	L	M
A	22	1	0	0	1	0	0	0	0	0	0	0	0	0
B	0	21	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	22	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	22	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	16	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	22	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	22	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	22	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	22	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	22	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	22	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	22	0	0
M	0	0	0	0	5	0	0	0	0	0	0	0	0	22

Overall Statistics

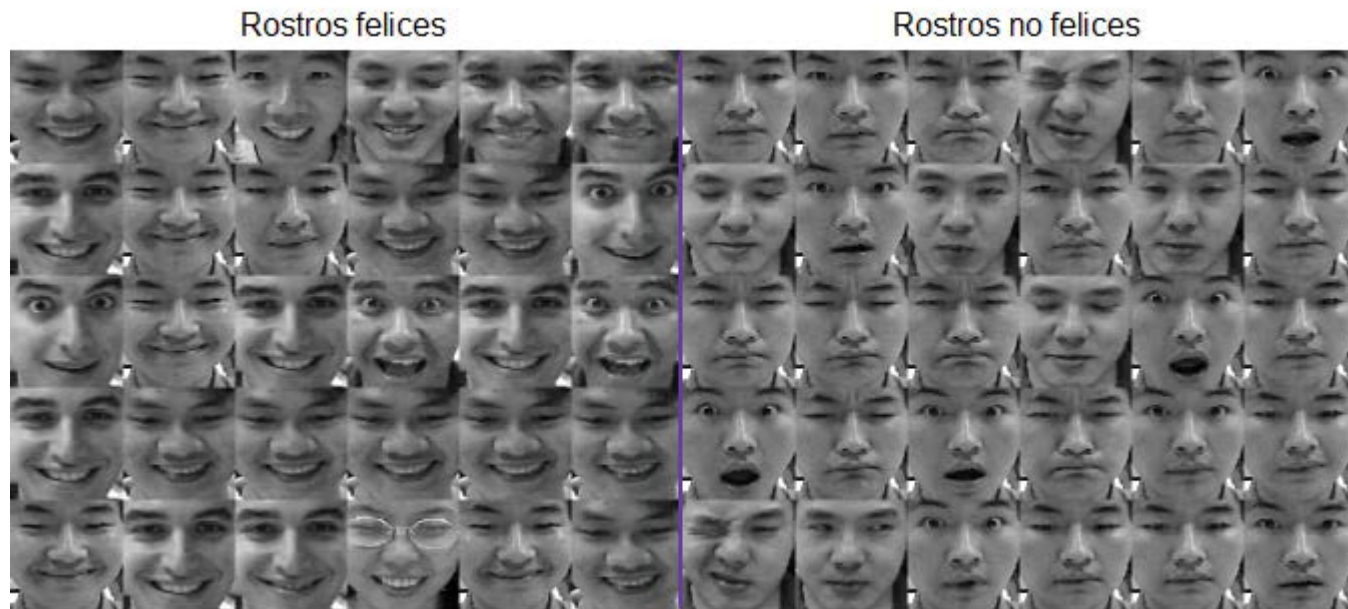
Accuracy : 0.9755  
 95% CI : (0.9502, 0.9901)  
 No Information Rate : 0.0769  
 P-Value [Acc > NIR] : < 2.2e-16  
  
 Kappa : 0.9735





### 3. Aprendizaje no supervisado: SOM

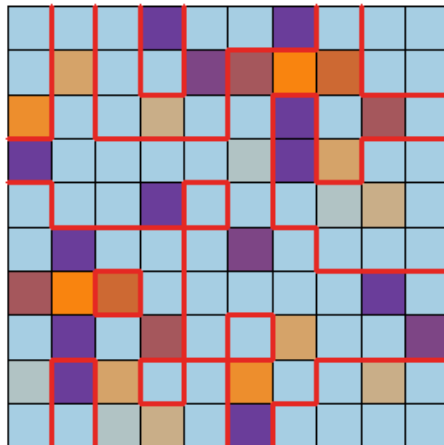
- Otros ejemplos:
- Reconocimiento expresión facial



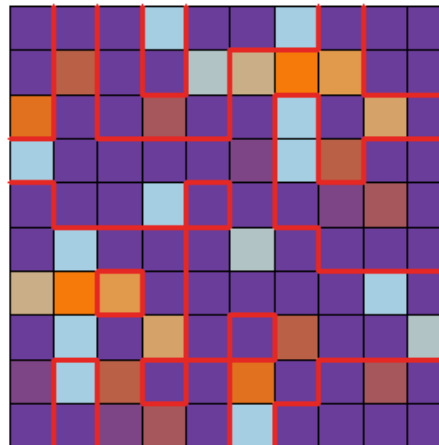
# 3. Aprendizaje no supervisado: SOM

- Otros ejemplos:
- Reconocimiento expresión facial

No feliz



Feliz



Confusion Matrix and Statistics

		Reference	
Prediction		0	1
	0	224	20
	1	9	33

Accuracy : 0.8986  
95% CI : (0.8576, 0.931)

No Information Rate : 0.8147  
P-Value [Acc > NIR] : 6.7e-05

Kappa : 0.6349

# Índice de contenidos

---

## 1. Introducción

## 2. Aprendizaje supervisado

- K-NN
- SVM
- Redes Neuronales: Perceptrón

## 3. Aprendizaje no supervisado

- K-means
- Redes Neuronales: SOM

## 4. Deep learning: Yolo

## 4. Deep learning: Yolo

---

- Cuando se trata de la detección de objetos basada en el aprendizaje profundo, hay tres detectores de objetos principales:
  - R-CNN y sus variantes, incluido el R-CNN original, Fast R-CNN y Faster R-CNN
  - Single Shot Detector (SSD): Detector de disparo único
  - YOLO
- Los R-CNN son uno de los primeros detectores de objetos basados en aprendizaje profundo y son un ejemplo de un detector de dos etapas:
  - En la **primera aproximación** (Girshick et al., 2013) se propuso un detector de objetos que requería un algoritmo como la búsqueda selectiva (o equivalente) para proponer Bounding boxes candidatos que pudieran contener objetos
  - Luego, estas regiones se pasan a una CNN para su clasificación, lo que finalmente condujo a uno de los primeros detectores de objetos basados en aprendizaje profundo
- El problema con el método estándar de R-CNN (Region Based Convolutional Neural Networks) era que era tremendamente lento y no un detector de objetos completo de un extremo a otro

## 4. Deep learning: Yolo

---

- Después, en 2015 Girshick et al. propuso el algoritmo Fast R-CNN, el cual realizó mejoras considerables en el R-CNN original, es decir, **aumentó la precisión y redujo el tiempo** necesario para realizar un pase hacia adelante; sin embargo, el modelo aún se basaba en un algoritmo de propuesta de región externa
- No fue hasta el artículo de seguimiento de 2015, que las R-CNN se convirtieron en un verdadero detector de objetos de aprendizaje profundo de extremo a extremo por eliminar el requisito de búsqueda selectiva y, en su lugar, confiar en una red de propuesta de región (RPN) que es (1) **totalmente convolucional** y (2) **puede predecir los cuadros delimitadores de objetos y las puntuaciones de "objetividad"** (es decir, una puntuación que cuantifica la **probabilidad** de que sea una región de una imagen puede contener una imagen). Las salidas de los RPN se pasan luego al componente R-CNN para su clasificación y etiquetado final

## 4. Deep learning: Yolo

---

- Si bien las **R-CNN** tienden a ser **muy precisas**, el mayor problema con la familia de redes R-CNN es su velocidad: eran **increíblemente lentas**, obteniendo solo 5 FPS en una GPU
- Para ayudar a aumentar la velocidad de los detectores de objetos basados en el aprendizaje profundo, tanto los detectores de disparo único (SSD) como YOLO, utilizan una estrategia de detector de una etapa
- Estos algoritmos tratan la detección de objetos como un **problema de regresión**, tomando una imagen de entrada dada y aprendiendo simultáneamente las coordenadas del cuadro delimitador y las probabilidades de etiqueta de clase correspondientes
- En general, los **detectores de una etapa** tienden a ser menos precisos que los **detectores de dos etapas**, pero son significativamente más rápidos
- YOLO es un gran ejemplo de detector de una sola etapa

## 4. Deep learning: Yolo

---

- Fue en 2015 que Redmon et al. detalla un detector de objetos capaz de detectar objetos en tiempo súper real, obteniendo 45 FPS en una GPU
  - Nota: Una variante más pequeña de su modelo llamada "Fast YOLO" afirma lograr 155 FPS en una GPU
  - YOLO ha pasado por varias iteraciones diferentes, incluido YOLO9000: Mejor, más rápido, más fuerte (es decir, YOLOv2), capaz de detectar más de 9,000 detectores de objetos
- Redmon y Farhadi pueden lograr una gran cantidad de detecciones de objetos al realizar un entrenamiento conjunto tanto para la detección como para la clasificación de objetos. Mediante el entrenamiento conjunto, los autores entrenaron a YOLO9000 simultáneamente en el conjunto de datos de clasificación de ImageNet y el conjunto de datos de detección de COCO. El resultado es un modelo YOLO, llamado YOLO9000, que puede predecir detecciones para clases de objetos que no tienen datos de detección etiquetados

## 4. Deep learning: Yolo

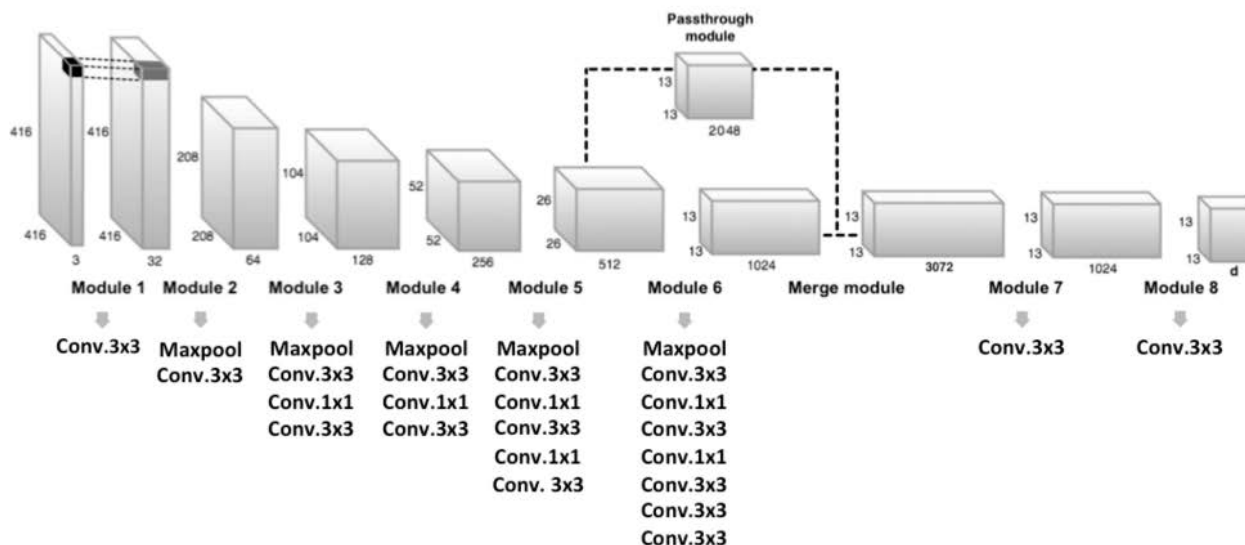
---

- Si bien es interesante y novedoso, el desempeño de YOLOv2 fue un poco decepcionante dado el título y el resumen del artículo
- De las 156 clases de COCO, YOLO9000 logró un 16% de precisión promedia (mAP), y sí, mientras que YOLO puede detectar 9000 clases separadas, la precisión no es exactamente la que desearíamos
- Fue en 2018 cuando Redmon y Farhadi publicaron YOLOv3, el cual es significativamente más grande que los modelos anteriores pero, es el mejor de la familia de detectores de objetos YOLO
- El conjunto de datos de COCO consta de 80 etiquetas, que incluyen, entre otras: Personas, Bicicletas, Coches y Camiones, Aviones, Señales STOP y bocas de incendio, Animales (incluidos gatos, perros, pájaros, caballos, vacas y ovejas, ...), Objetos de cocina y comedor (como copas de vino, tazas, tenedores, cuchillos, cucharas, etc....) ¡y mucho más!



# 4. Deep learning: Yolo

- Toda la imagen se introduce en el modelo. Esa es esencialmente la razón por la que YOLO es tan rápido. Mira la imagen completa solo una vez
- Esto lo hace la CNN. Básicamente, cada porción de una convolución corresponde a una celda de cuadrícula. Por ejemplo, la celda superior derecha de una imagen correspondería a la parte superior derecha de los filtros de cada capa. Esto se visualiza a la izquierda de esta imagen:



# 4. Deep learning: Yolo

