Recuperación de Información Multimedia

Repaso Machine Learning

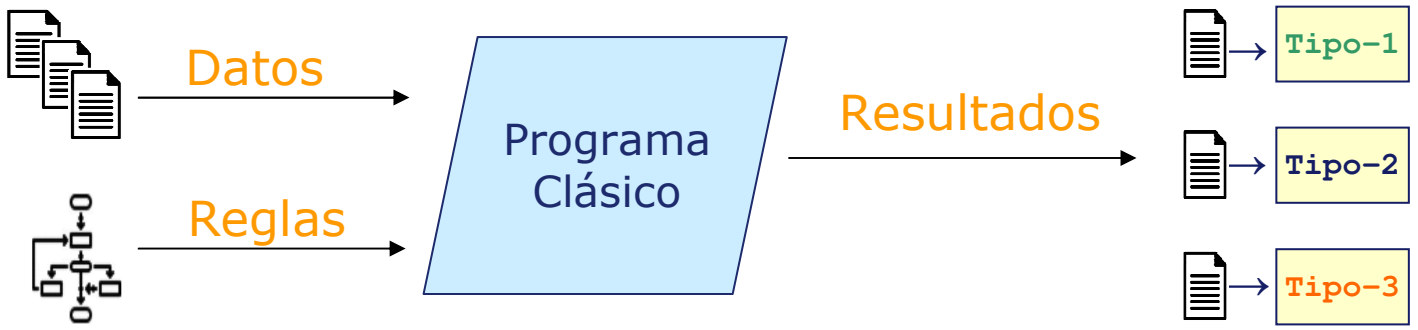
CC5213 – Recuperación de Información Multimedia
Departamento de Ciencias de la Computación
Universidad de Chile
Juan Manuel Barrios – https://juan.cl/mir/ – 2019

Inteligencia Artificial

- Se refiere a los esfuerzos por “automatizar tareas intelectuales normalmente realizadas por humanos”
- Enfoques:
  - Neurobiología
  - Symbolic AI (lógica+reglas)
  - Machine Learning (datos+estadística)

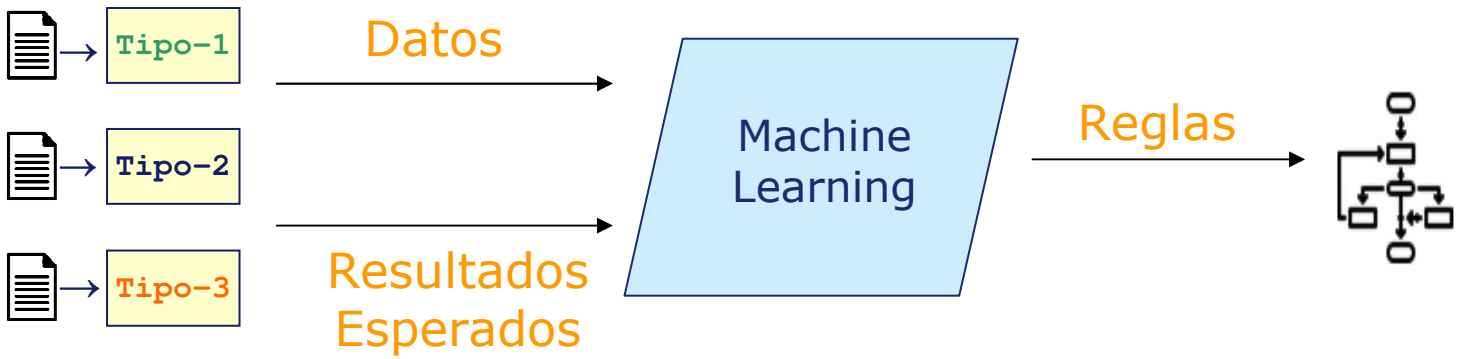
Programación Clásica

- Aplicar reglas sobre datos para generar una salida deseada



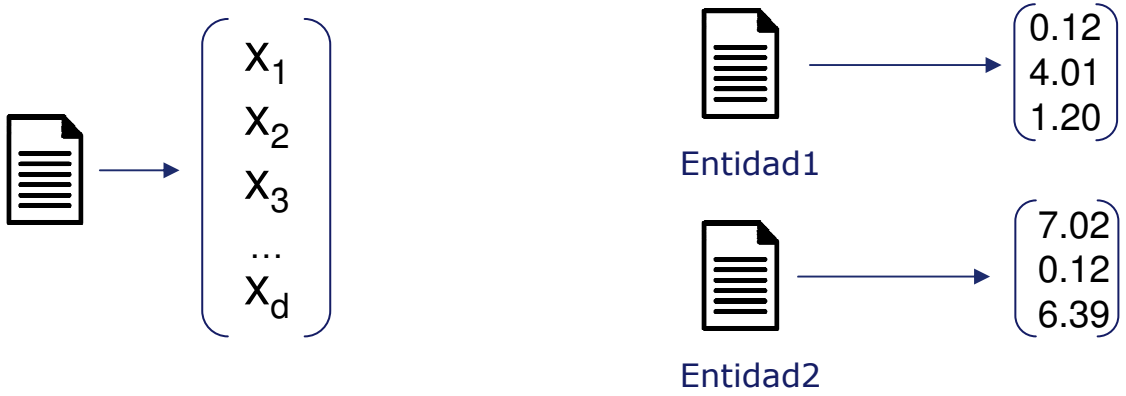
Machine Learning

- Aprender o descubrir las reglas que permiten producir el resultado deseado



Representación del Contenido

- Se tiene un conjunto de **entidades** que se desea analizar. Ej: clientes, productos, e-mails, páginas web, fotos, canciones, etc.
- Cada **entidad** se debe representar por sus datos, usualmente modelados como un vector de dimensión fija (vector en R^n)
  - Feature Vector, Vector Característico, Descriptor



Machine Learning

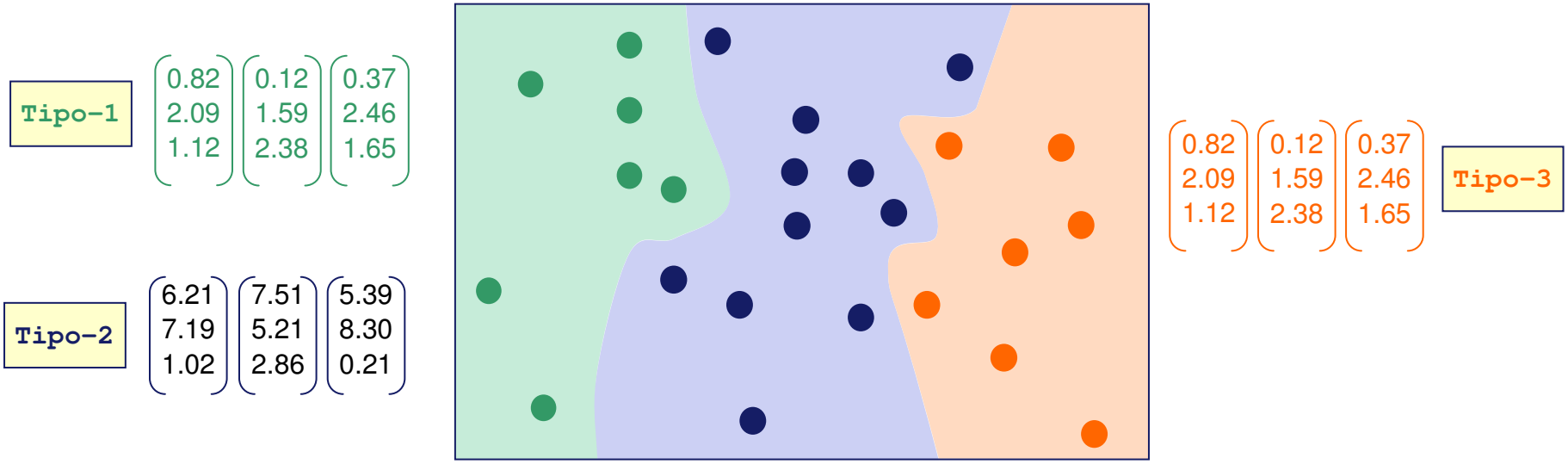
- Se tienen entidades (representadas por sus descriptores) y se desea encontrar reglas o patrones
- Métodos Supervisados:
  - Se tienen datos etiquetados (entidades con la respuesta correcta)
  - Tareas comunes: Asignar etiquetas (clasificación, caso discreto). Generar uno o más números (regresión, caso continuo).
- Métodos No Supervisados:
  - No hay información extra (a parte de los descriptores)
  - Tarea común: Encontrar grupos de descriptores similares e interpretar esos grupos (clustering).

Clasificadores

- Se tiene un conjunto clases o etiquetas $C=\{C_1,..,C_n\}$
- Se tiene uno o más descriptores de ejemplo para cada clase C_i
- Se desea asignar la etiqueta que le corresponde a entidades nuevas (ej. etiquetar tipos de correos, etiquetar tipos de clientes, etc.)
- La etapa de entrenamiento consiste en encontrar el patrón de cada clase
- La etapa de clasificación consiste en etiquetar descriptores nuevos

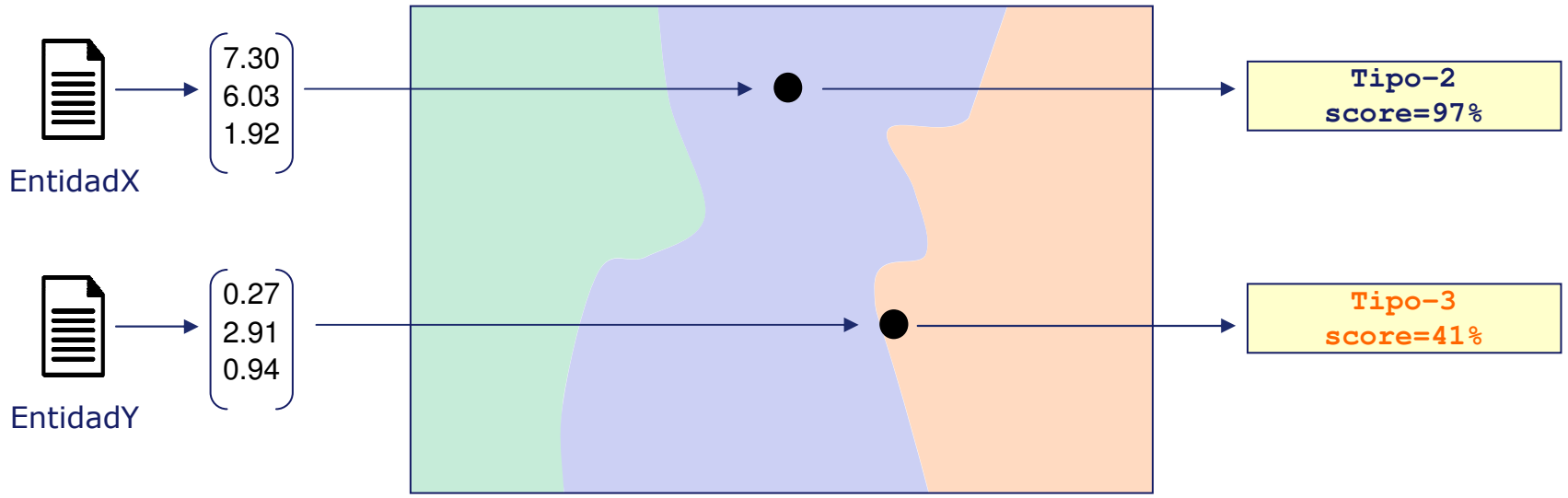
Entrenamiento de Clasificador

- Se tienen descriptores de las entidades y sus etiquetas
- Se desea determinar una función *F* que asocie cada descriptor con su etiqueta $F: R^n \rightarrow \{\text{etiqueta}_1, \ldots, \text{etiqueta}_n\}$
  - *F* busca regiones de R^n con descriptores de una misma etiqueta
  - *F* determina las fronteras de separación entre esas regiones



Uso de Clasificador

- Clasificación de una **entidad nueva**:
  - Obtener o calcular el descriptor que corresponde a esa entidad
  - Utilizar el clasificador ya entrenado para determinar la **región** a la que pertenece el descriptor
  - Obtener la **etiqueta** asociada a esa zona junto con un **valor de confianza** de que sea la etiqueta correcta



Entrenamiento y Underfit

- Los clasificadores se ajustan (entrenan) automáticamente mediante un algoritmo que usa todos los datos de entrenamiento
- Los clasificadores usualmente tienen además parámetros de mayor nivel (hiperparámetros) que se deben ajustar manualmente
  - Se debe probar varias veces con distintos hiperparámetros hasta lograr que el entrenamiento logre un buen resultado de clasificación
- **Underfit** ocurre cuando no se logra un buen resultado de clasificación sobre los datos de entrenamiento
  - Los datos son muy difíciles de agrupar en regiones
  - Se deben ajustar los hiperparámetros para aumentar el “poder expresivo” del clasificador

Datos de Producción

- Usualmente se desea entrenar un clasificador para etiquetar datos nuevos (datos de producción)
  - Los datos usados para el entrenamiento deben ser similares a los que después se usarán en producción
- **Overfit (1)** ocurre cuando se obtiene una gran diferencia en los resultados de clasificación entre los datos de **entrenamiento** y los de **producción**
  - El clasificador funciona bien con datos conocidos, pero al instalarlo en producción y clasificar datos nuevos se obtienen malos resultados
  - **Problema grave**, el proyecto tiene altas probabilidades de fracasar
  - Para poder estimar el resultado que se obtendrá con datos futuros, antes de entrenar se separa una fracción datos (conjunto de test)

Datos de Test

- Los datos de test son una muestra de los datos de producción
  - Permiten estimar el resultado que obtendrá el clasificador al instalarlo en producción y darse cuenta si ocurrirá un Overfit (1)
  - Si se usan para mejorar el clasificador puede ocurrir un Overfit (1)
- **Overfit (2)** ocurre cuando se obtiene una gran diferencia en los resultados de clasificación entre los datos de **entrenamiento** y los de **test**
  - El algoritmo de entrenamiento memoriza datos pero no encuentra un patrón en los datos
  - Para ajustar hiperparámetros se deben hacer muchos experimentos de clasificación
    - Si se prueban muchos clasificadores y se escoge el que logra el mejor resultado en test se puede producir Overfit (1)
    - Crear un conjunto distinto a test llamado validación

Datos de Validación

- El conjunto de validación permite probar distintos hiperparámetros
  - Son parte de los datos usados para entrenar
  - Puede ser un subconjunto fijo o un subconjunto al azar
- **Overfit (3)** ocurre cuando se obtiene una gran diferencia en los resultados de clasificación entre los datos de **entrenamiento** y los de **validación**
  - Detener el entrenamiento por épocas cuando difieren mucho los resultados entre entrenamiento y validación
  - Validación cruzada (cross validation, cross-folding) se hace rotar el conjunto de validación y se calcula el resultado promedio

Clasificadores

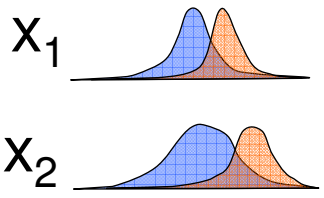
- Los distintos clasificadores se diferencian en el enfoque usado para procesar descriptores y crear regiones
- Ejemplos de clasificadores:
  - Clasificador Bayesiano
  - Árbol de Decisión
  - Clasificador K-NN
  - Máquina de vectores de soporte (SVM)
  - Redes Neuronales Artificiales

Clasificador Bayesiano

- Con datos de entrenamiento se obtienen las distribuciones de probabilidad de los descriptores de cada clase
- Para una entidad nueva, se calcula su descriptor y se calcula la clase más probable utilizando el Teorema de Bayes:

$$P(\text{Clase} \mid \vec{x}) = \frac{P(\vec{x} \mid \text{Clase})\ P(\text{Clase})}{P(\vec{x})}$$

Entrenamiento:



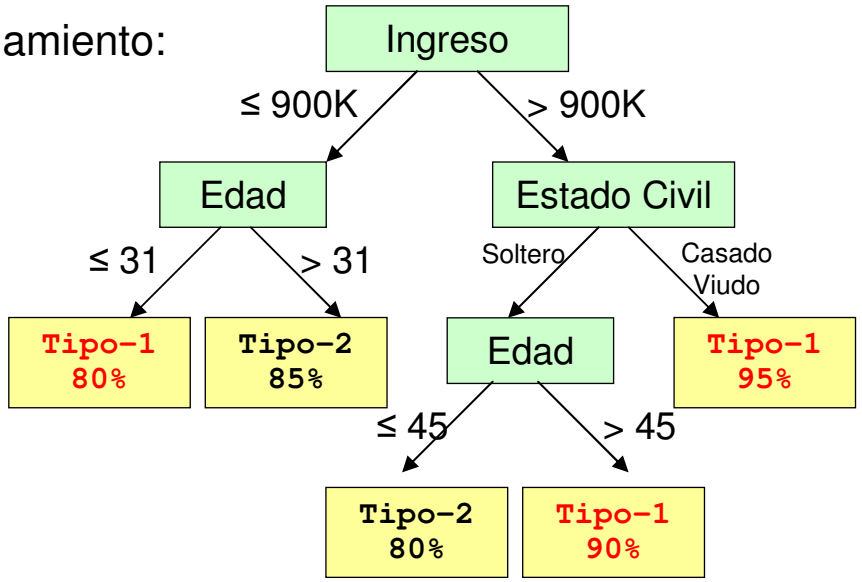
$$\vec{x} = (152, 350\text{K})$$

$$P(\texttt{Tipo1} \mid \vec{x}) = P(\vec{x} \mid \texttt{Tipo1})\ P(\texttt{Tipo1})$$
$$P(\texttt{Tipo2} \mid \vec{x}) = P(\vec{x} \mid \texttt{Tipo2})\ P(\texttt{Tipo2})$$

Árbol de Decisión

- Usar los descriptores de entrenamiento para determinar una secuencia de atributos y rangos que mejor separen los descriptores en cada clase
- Random Forest: Se calculan varios árboles sobre los mismos datos

Entrenamiento:



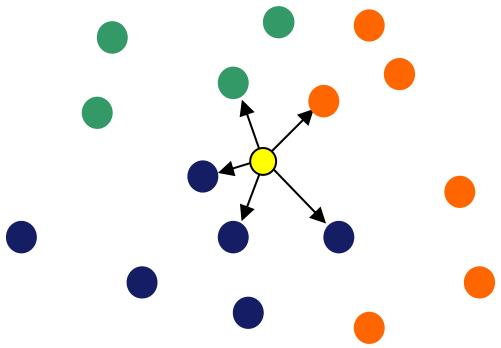
$$\vec{x} = (38, 950\text{K}, \texttt{Soltero})$$

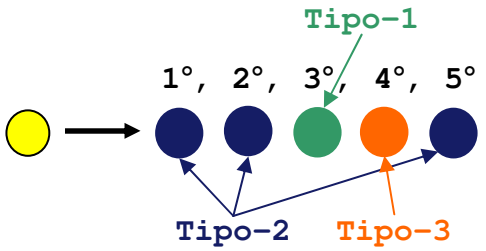
Tipo-2
score=80%

Clasificador k-NN

- El entrenamiento consiste en indexar vectores (Linear Scan, R-tree, kd-tree, kmeans tree, LSH, etc.)
- Para clasificar un descriptor nuevo, se buscan los *k* descriptores de entrenamiento más cercanos
- Se realiza una votación entre las *k* etiquetas de entrenamiento encontradas y se decide la más votada

Entrenamiento:





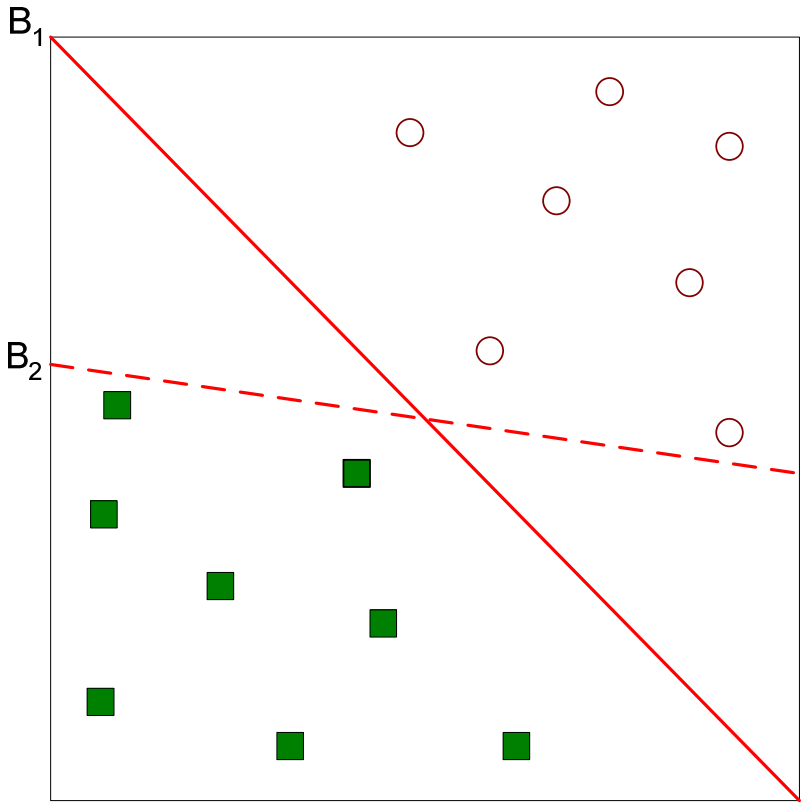
Resultado: Tipo-2 score=60%

Clasificador k-NN

- Métodos de votación de etiquetas:
  - El voto del *i*-ésimo NN vale *k*+1-*i*
  - El voto del *i*-ésimo NN vale 1/i
  - El voto del NN a distancia *d* vale 1/*d*
- No generaliza a regiones (Lazy Learning)
- Basta un ejemplo por clase
- Si se usa un índice dinámico (ej: R-tree) se pueden actualizar los datos de entrenamiento online
- Difícil escalar con muchos datos de entrenamiento
- Es posible explicar cada respuesta (mostrar los datos de entrenamiento que causaron una respuesta)

Support Vector Machine (SVM)

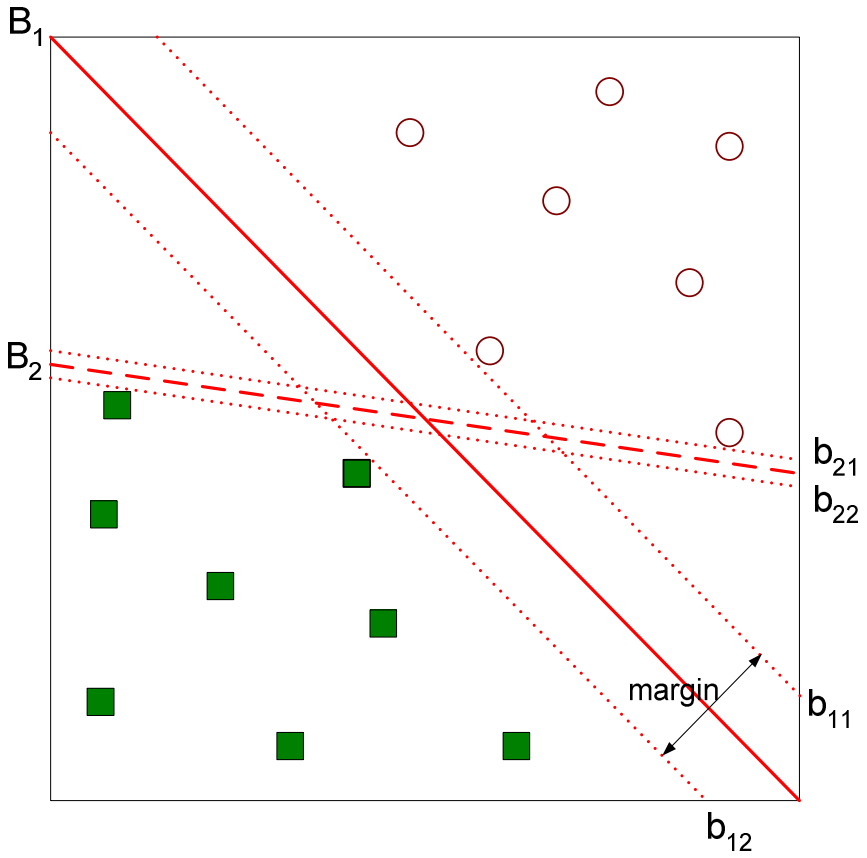
- Clasificador lineal de vectores en el espacio
- Se basa en calcular un hiperplano que separa las dos clases
- Dado un conjuntos de datos de entrenamiento, encontrar la recta que separa las dos clases



Fuente: http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap5_alternative_classification.ppt

SVM

- Existen muchos planos que pueden separar las dos clases
- Elegir el plano que maximiza el “margen”



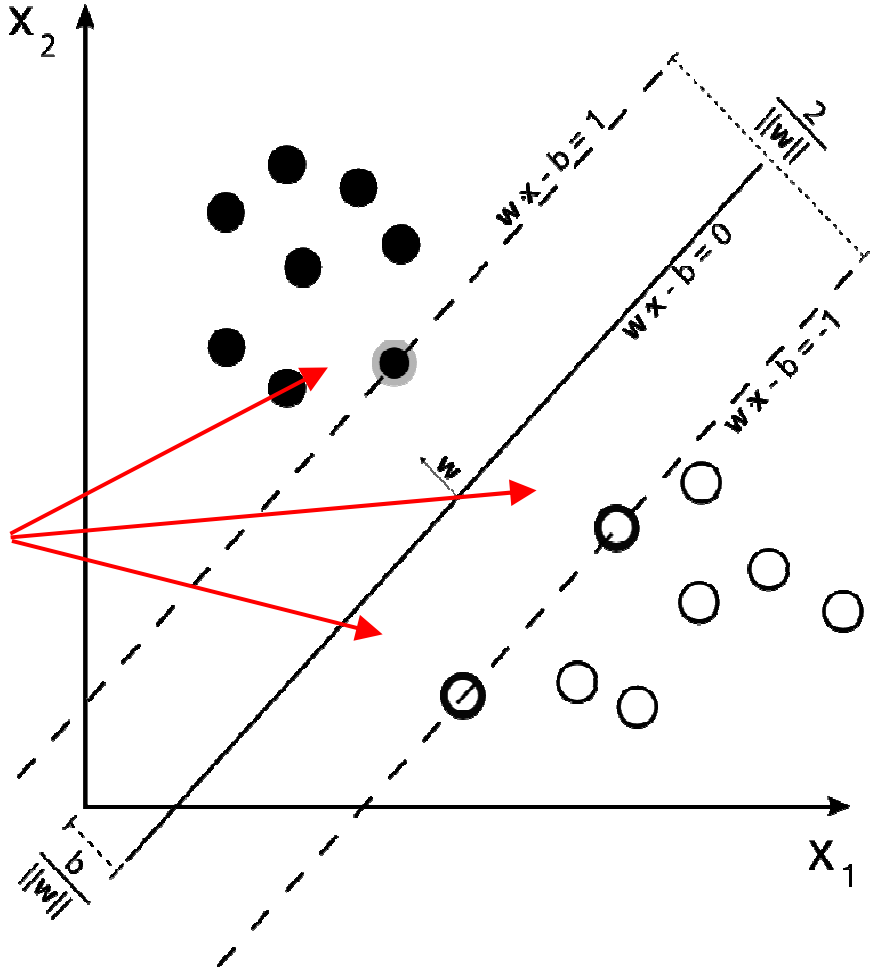
Fuente: http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap5_alternative_classification.ppt

SVM

- Un hiperplano es definido como todos los puntos $\vec{x}$ del espacio que cumplen:

$$\vec{w} \cdot \vec{x} - b = 0$$

- Donde $\vec{w}$ es la normal y es la $b$istancia al plano.

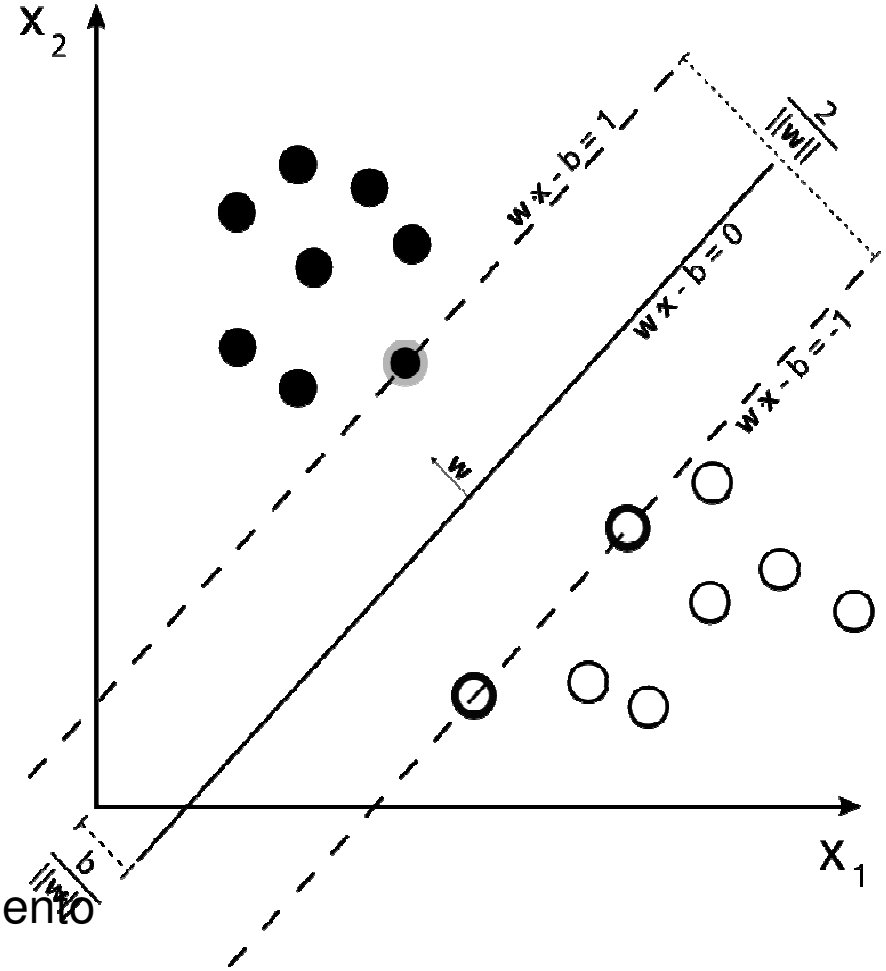


SVM

- Cuando un punto está en la frontera de la clase 1 entonces $\vec{w} \cdot \vec{x} - b$ vale 1, y si está en la otra frontera vale -1
- Se desea encontrar el plano con mayor distancia entre las dos fronteras (margen):

$$\min_{\vec{w},b} \frac{1}{2}||\vec{w}||^2$$

$$\text{sujeto a} \qquad c_i(\vec{w} \cdot \vec{x_i} - b) \geq 1$$



Donde x_i es el i-ésimo dato de entrenamiento que pertenece a la clase c_i (-1 o 1)

SVM: Soft Margin

- Las clases podrían no ser linealmente separables o pueden existir datos de entrenamiento incorrectos
- Se usa un “Soft Margin” para asignar una penalización a los datos incorrectamente clasificados:

$$\min_{\vec{w},b} \ \frac{1}{2}||\vec{w}||^2 + C\sum_i \xi_i$$

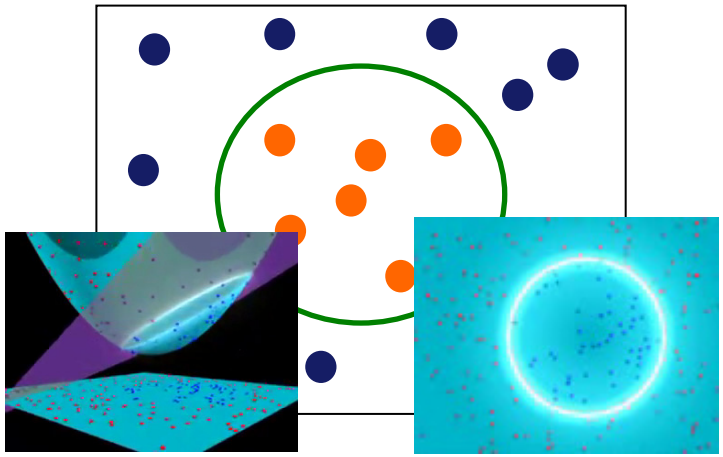
$$\text{sujeto a} \qquad c_i(\vec{w}\cdot\vec{x_i} - b) \geq 1 - \xi_i$$

- Donde *C* es un parámetro y ξ_i es el grado de error permitido al dato x_i
- Al resolver este problema con multiplicadores de Lagrange la solución final sólo depende de *C* que queda como parámetro de clasificador

SVM: Kernel Trick

- SVM sólo puede separar clases linealmente, pero es común requerir separaciones no lineales
- Solución: modificar el espacio para llevarlo a una dimensión mayor usando una función especial:
- El producto punto $\vec{w} \cdot \vec{x}$ se reemplaza por una función no lineal y se usa el SVM lineal en el nuevo espacio
- Kernels usuales:
  - Lineal $k(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y}$
  - RBF $k(\vec{x}, \vec{y}) = e^{\frac{-||\vec{x}-\vec{y}||^2}{std^2}}$
  - Polinomial $k(\vec{x}, \vec{y}) = (s(\vec{x} \cdot \vec{y}) + r)^d$

http://youtu.be/3liCbRZPrZA



SVM: Multiclase

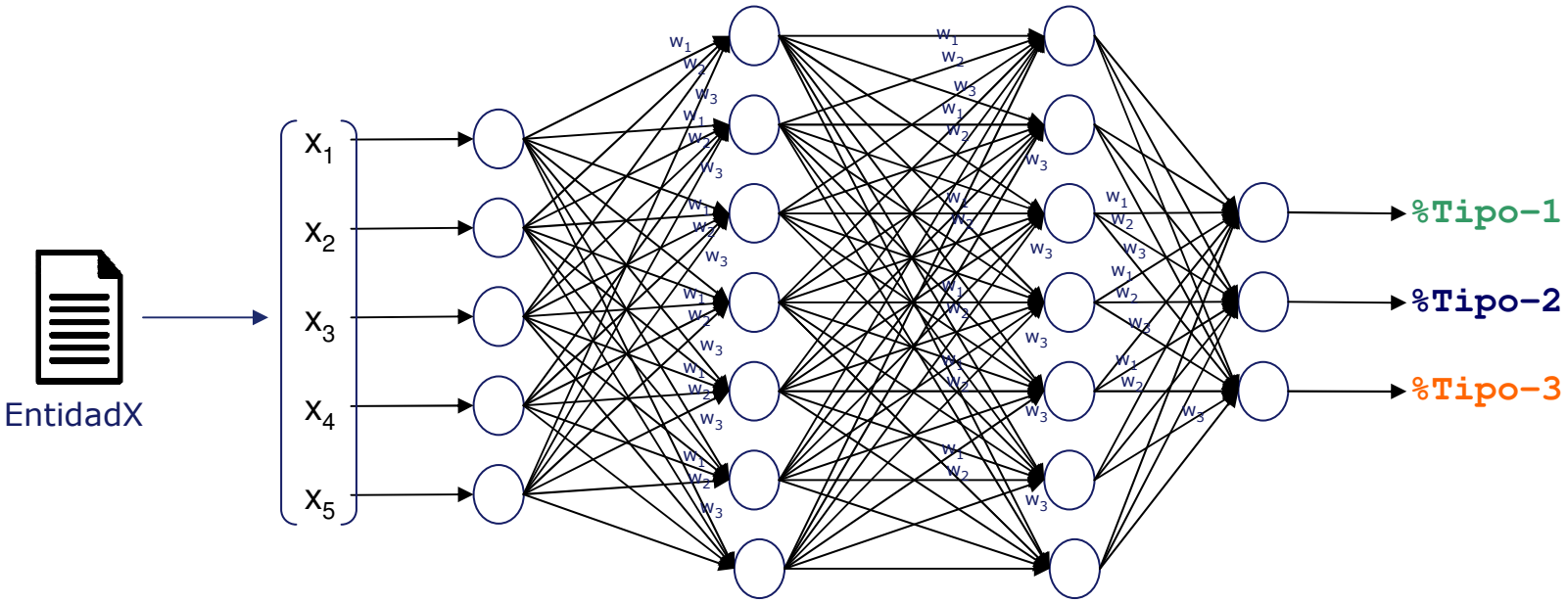
- SVM es un clasificador binario:

$$\vec{u} \in \omega_1 \Leftrightarrow h(\vec{u}) \geq 0$$
$$\vec{u} \in \omega_2 \Leftrightarrow h(\vec{u}) < 0$$

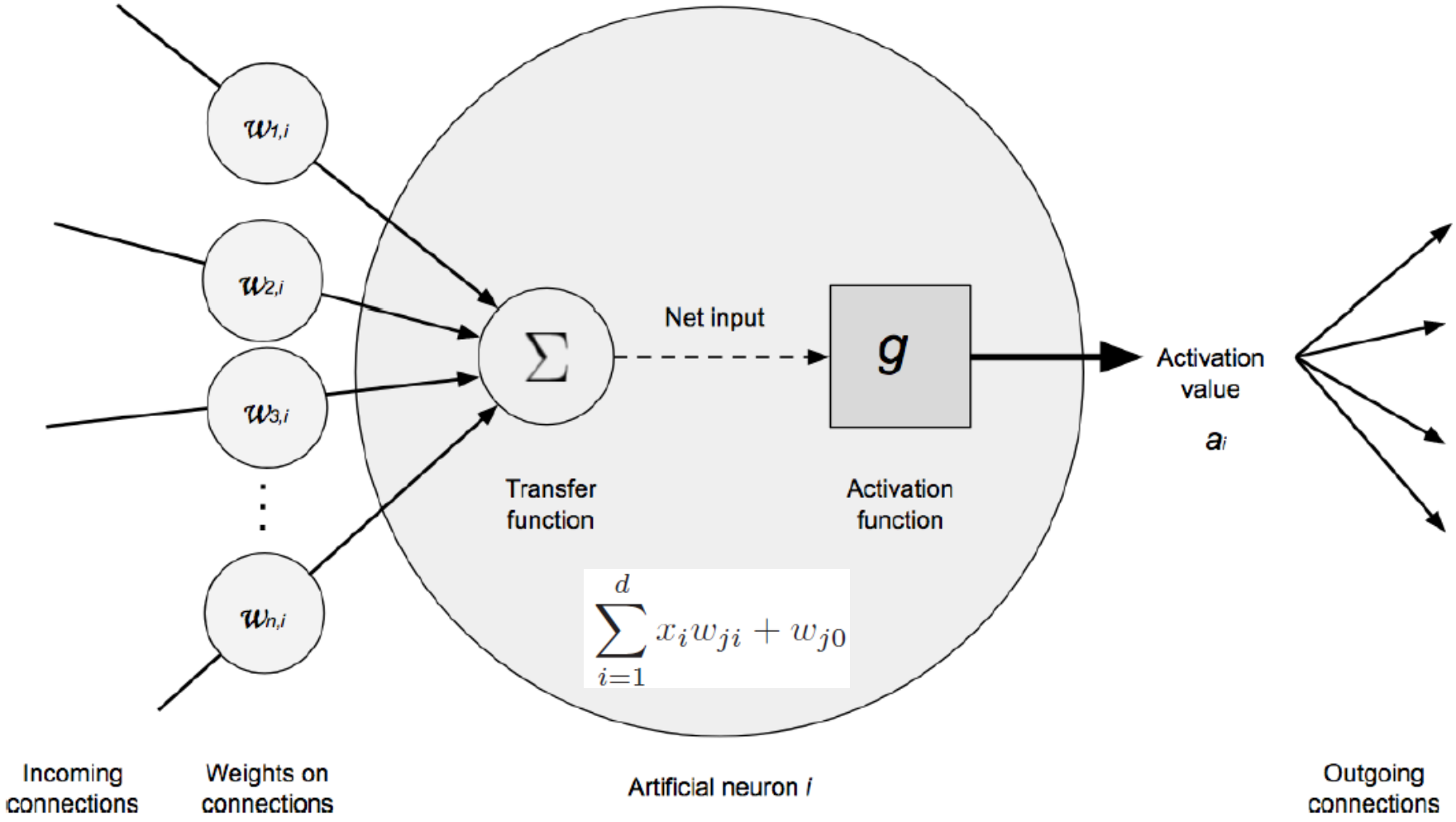
- “one-versus-all”: usar un SVM para cada clase, evaluar cada uno de ellos y elegir el de mayor confianza
- “one-versus-one”: entrenar un SVM para cada par de clases, evaluar en cada uno de ellos y elegir la clase con mayor votación

Red Neuronal Artificial

- Redes Neuronales Artificiales (MLP)
  - La función de clasificación se modela como operaciones en capas
  - Capa de **entrada**, capas **escondidas** y capa de **salida**
  - El **entrenamiento** ajusta los **pesos internos** que logran **convertir** cada **entrada** en la **salida** deseada



MLP: Perceptrón

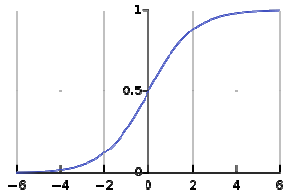


MLP: Parámetros

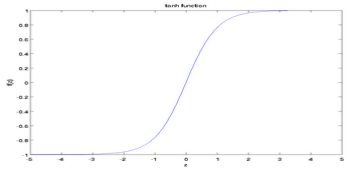
- Parámetros: pesos de las conexiones entre neuronas
- Función objetivo: error entre la salida de la red y la salida esperada
- Hiperparámetros: arquitectura de la red, capas, cantidad de neuronas, funciones de activación, etc.
- Entrenamiento por Back Propagation. Ver videos:
  - Parte 1: https://www.youtube.com/watch?v=aircAruvnKk
  - Parte 2: https://www.youtube.com/watch?v=IHZwWFHWa-w
  - Parte 3: https://www.youtube.com/watch?v=Ilg3gGewQ5U

MLP: Función de Activación

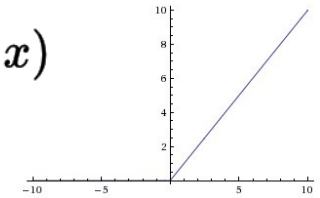
- Logistic (Sigmoid) $f(x) = \frac{1}{1 + e^{-x}}$



- Tanh $tanh(x) = \frac{2}{1+e^{-2x}} - 1$



- Rectified Linear Unit (ReLU) $f(x) = \max(0, x)$



- Leaky ReLU $f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$

- Parametric ReLU
  - Reemplazar la constante 0.01 por un parámetro

MLP: Entrenamiento

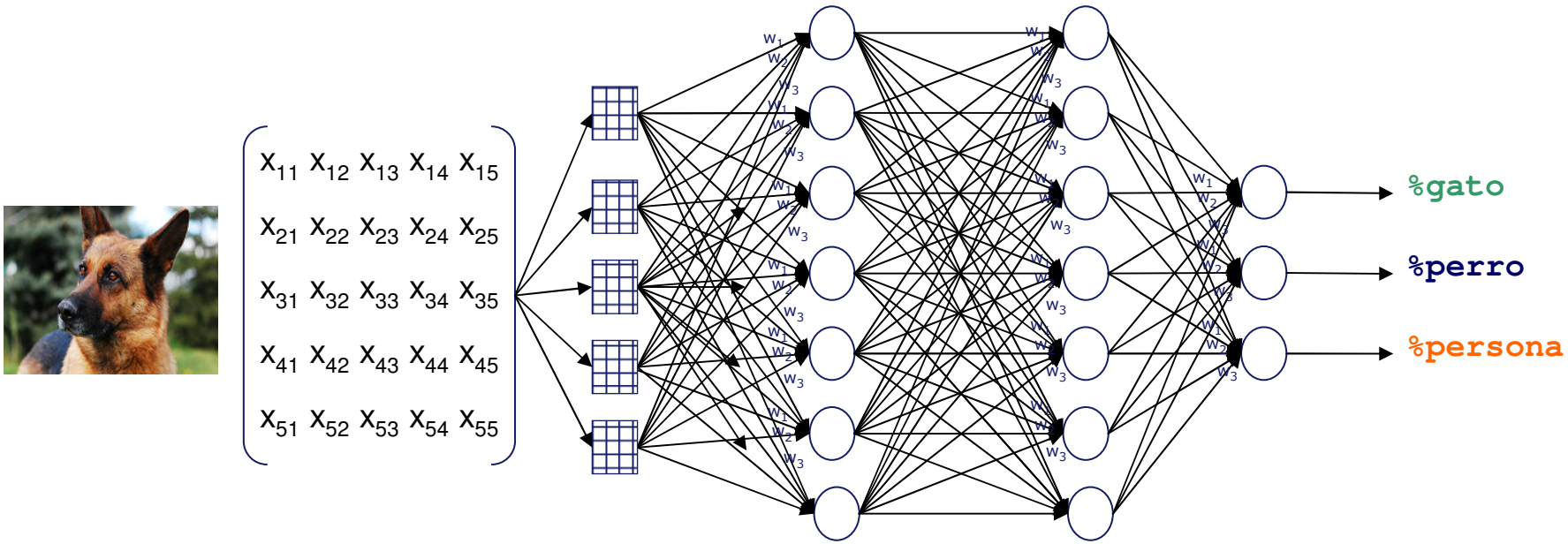
- Mini-batches: Entrenar por subconjuntos de vectores
- Stochastic Gradient Descent: El cálculo se realiza por grupos aleatorios (Stochastic=Random)
- Learning Rate: Ponderación a la modificación de pesos
- Dropout: En cada ciclo de entrenamiento se desactiva una fracción de las conexiones (usualmente 20-50%). Evita overfit al no permitir conexiones fuertes entre perceptrones
- Regularization: A la función de costo se le agrega el valor numérico del peso. Preferir varios pesos pequeños a un peso grande.
- Data Augmentation: Generar más datos de entrenamiento con pequeñas transformaciones o perturbaciones
- Transfer Learning: Comenzar el entrenamiento con valores obtenidos de otros datasets (incluso otros dominios)

Deep Learning

- Se refiere al uso de redes neuronales “muy grandes” o “profundas”
- Se diferencia de uso de redes neuronales “tradicionales” (MLP) por:
  - Incluir el cálculo del descriptor de contenido en la misma red
    - La entrada de la red es el dato multimedia mismo
  - Las red contiene gran cantidad de parámetros (neuronas, capas, arquitecturas complejas)
    - Entrenamiento requiere gran poder computacional

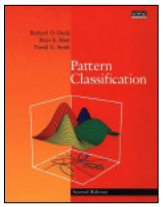
Red Neuronal Convolucional

- Red neuronal que contiene operadores de **convolución**
  - Forma eficiente cuando se buscan patrones con localidad espacial
  - El resultado de una convolución es una imagen de (casi) el mismo tamaño
  - Los valores del filtro son parámetros a entrenar
- El operador de pooling reduce el tamaño de una imagen
- El cálculo del descriptor es parte del entrenamiento

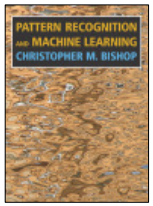


Bibliografía

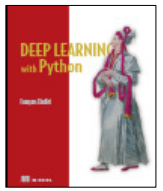
- **Pattern Classification. Second Edition**. Duda, Hart, Stork. 2001
  - Cap 6



- **Pattern Recognition and Machine Learning**. Bishop. 2006
  - Cap 5



- **Deep Learning with Python**. Chollet. 2018



https://xkcd.com/1838/