Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Sistemas Urbanos Inteligentes

Introducción a Redes Neuronales

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación

Retomemos el problema de estimación de tiempo de viaje, ahora con un foco distinto

- Hora del día
- Ubicación
- Inicio y fin (hora, ruta)
- Demanda según origen y hora
- Caracterización del instante
- Información reciente e histórica

Generalmente, podemos identificar la información que creemos necesaria para un problema

Retomemos el problema de estimación de tiempo de viaje, ahora con un foco distinto

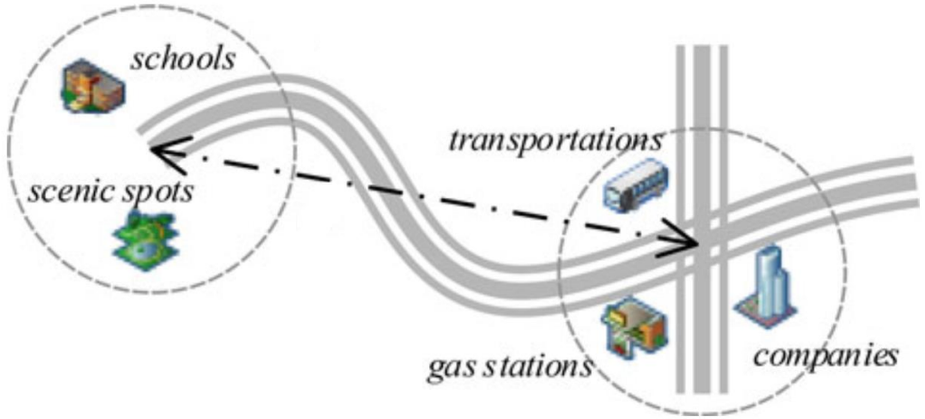


Figura tomada de Tang et al, "*Deep Architecture for Citywide Travel Time Estimation Incorporating Contextual Information*", JITS (2019)

- Ruta
- Modo – tipo vehículo
- Contexto ruta (figura izq.)
- GPS
- Bluetooth
- BIP (fare cards)
- Uso de suelo
- Clima
- Estado ruta
- Apps

Comúnmente, especificaríamos un modelo de regresión, que nos permita utilizar estos datos

Retomemos el problema de estimación de tiempo de viaje, ahora con un foco distinto

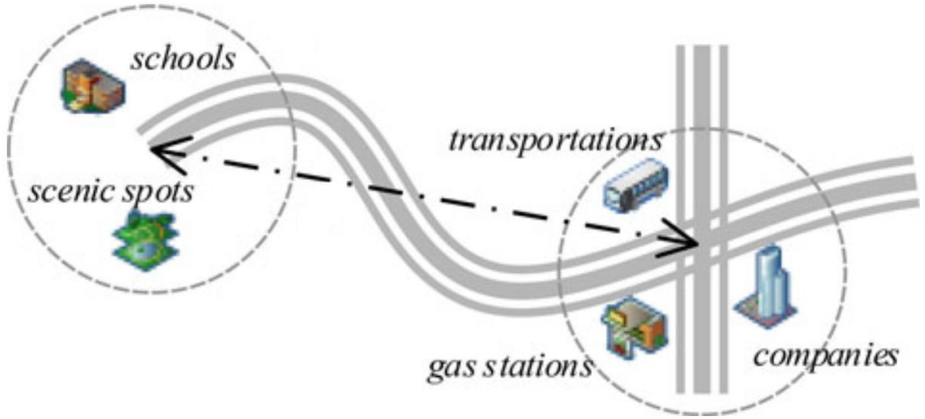


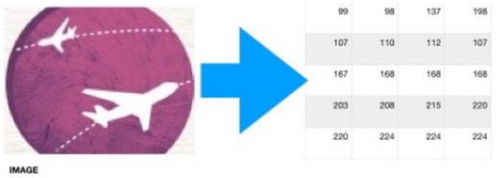
Figura tomada de Tang et al, "*Deep Architecture for Citywide Travel Time Estimation Incorporating Contextual Information*", JITS (2019)

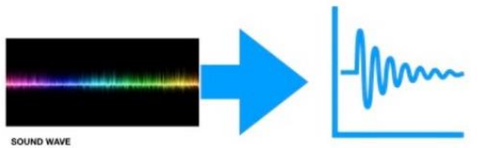
- Ruta
- Modo – tipo vehículo
- Contexto ruta (figura izq.)
- GPS
- Bluetooth
- BIP (fare cards)
- Uso de suelo
- Clima
- Estado ruta
- Apps

El problema es que, no siempre tenemos garantías de que este esquema nos permita representar la información de manera óptima

A partir de ahora, buscaremos utilizar la menor cantidad de especificaciones posibles

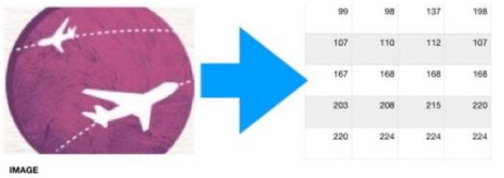
- Hasta ahora, hemos estado usando *features* y/o especificaciones basadas en modelos conocidos.
- Esto tiene que ver más con el conocimiento experto que tengamos (o no) del problema.
- Esto no es ideal por varias razones:
  i. Un experto es caro
  ii. Nada asegura que este conocimiento sea el mejor
  iii. Para datos no estructurados, no son evidentes las mejores features y/o especificaciones de modelo
  iv. No es siempre evidente como combinar distintos tipos de datos para extraer información

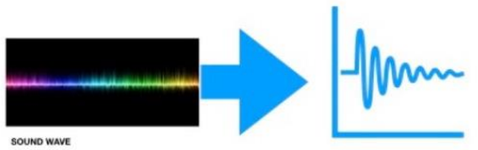




La clave para esto viene dada por las representaciones

- En vez de preguntarnos cómo modelar algo, nos preguntaremos cómo aprender algo.
- Pero esta pregunta no apuntará solamente a la tarea, como habíamos discutido hasta ahora.
- De ahora en adelante, nos preguntaremos cómo aprender representaciones, que nos permitan aprender de mejor manera cómo resolver la tarea.
- Disclaimer: no hay ninguna garantía de que podamos entender o interpretar estas representaciones.





Entonces, ¿cómo aprendemos estas representaciones?

- Para comenzar a responder esto, nos enfocaremos en dos puntos:

    1. Como estructurar el modelo de aprendizaje completo (representaciones + tarea)

    2. Cómo estructurar un esquema de aprendizaje de representaciones

- Revisaremos estos puntos, antes de llegar al primer modelo que discutiremos

Entonces, ¿cómo aprendemos estas representaciones?

- Para comenzar a responder esto, nos enfocaremos en dos puntos:

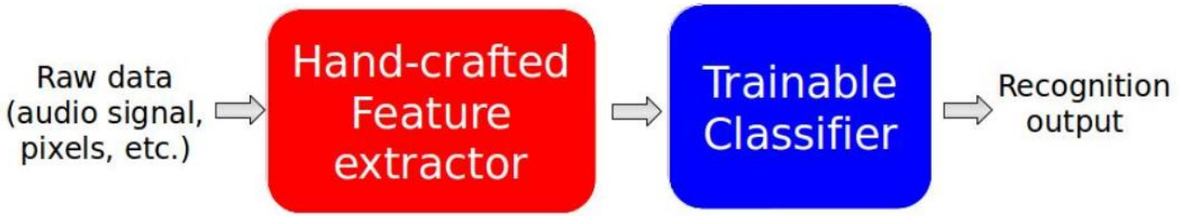
    1. Como estructurar el modelo de aprendizaje completo (representaciones + tarea)

    2. Cómo estructurar un esquema de aprendizaje de representaciones

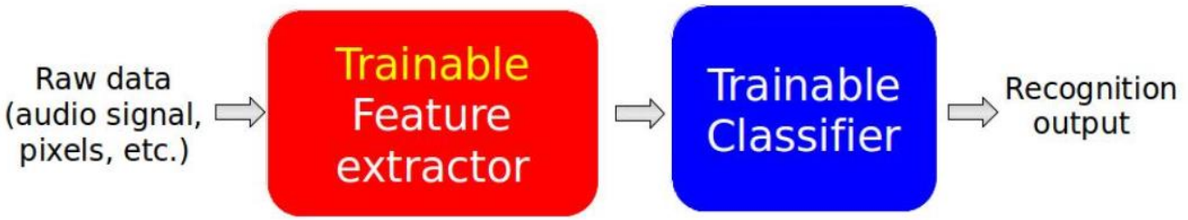
- Revisaremos estos puntos, antes de llegar al primer modelo que discutiremos

Enfoques modernos acoplan el aprendizaje de features al de la tarea

- El enfoque tradicional de reconocimiento de patrones se ha basado por 50 años en *features* hechas a mano:



- El enfoque moderno se basa en el aprendizaje de *features*:



Entonces, ¿cómo aprendemos estas representaciones?

- Para comenzar a responder esto, nos enfocaremos en dos puntos:

    1. Como estructurar el modelo de aprendizaje completo (representaciones + tarea)

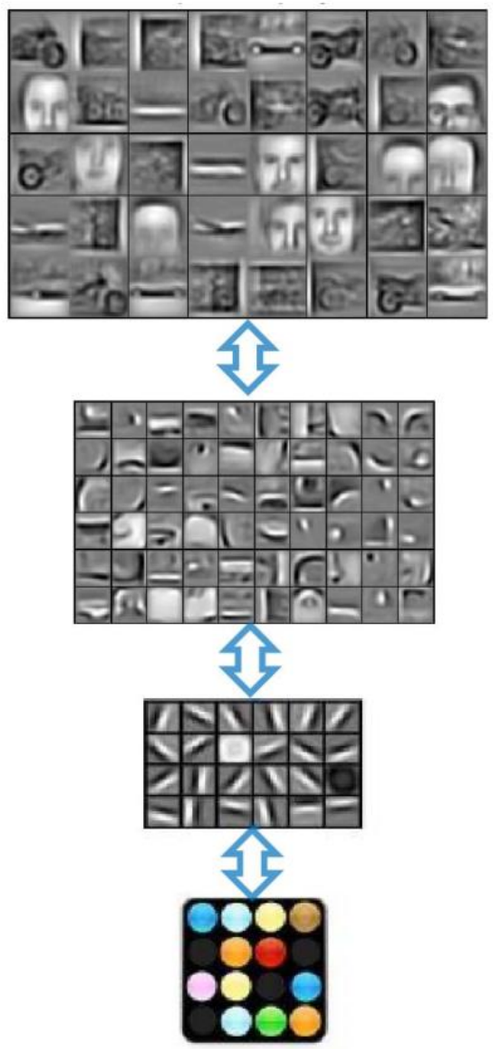
    2. Cómo estructurar un esquema de aprendizaje de representaciones

- Revisaremos estos puntos, antes de llegar al primer modelo que discutiremos

Representaciones composicionales y jerárquicas son muy comunes en la naturaleza

- Se organizan desde niveles bajos a altos de abstracción.
- La composicionalidad es su principal característica.
- Algunos ejemplos:
  - Sonidos -> fonemas -> sílabas -> palabras
  - Pixeles -> bordes -> partes -> objetos -> escenas
  - Caracteres -> palabras -> grupos -> clausulas -> frases
  - Nucleótidos -> genes -> proteínas -> células ->órganos

En visión, en particular, estas representaciones son muy intuitivas

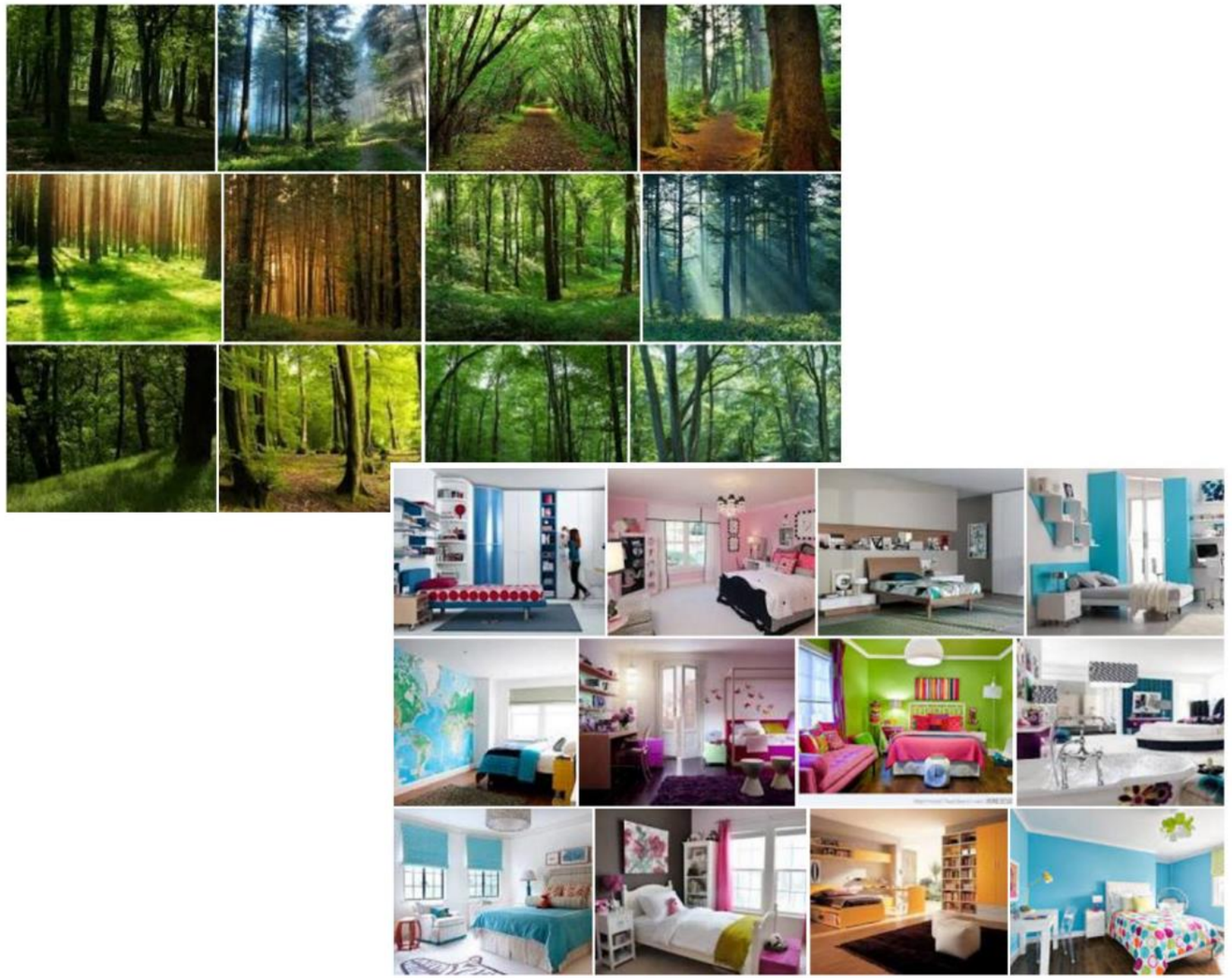


Intuitivamente, ¿qué podrían aprender los distintos niveles o capas?

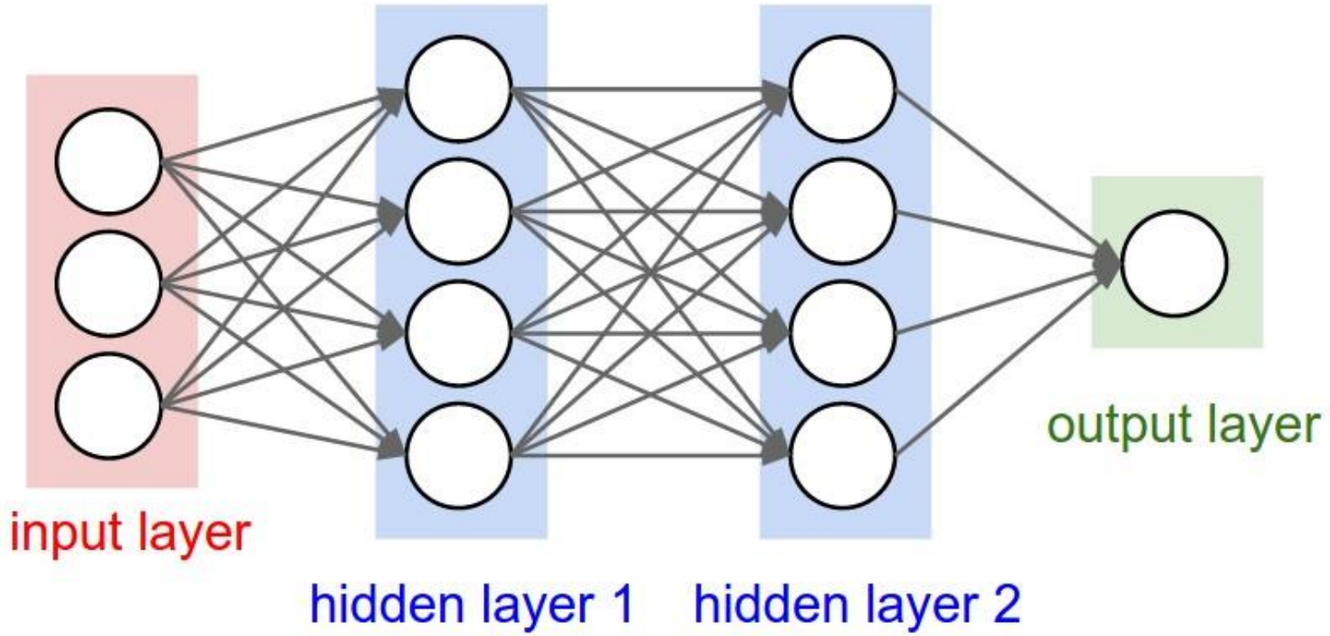
- Capas de bajo nivel (poca profundidad) aprender representaciones genéricas y locales que funcionan como bloques de construcción: fonemas y bordes en reconocimiento de voz y visual, respectivamente.

- Capas de alto nivel capturan representaciones especializadas y globales, que son más robustas a variaciones: independientes del que habla o del punto de vista, en reconocimiento de voz y visual, respectivamente.

Composicionalidad es clave para combinar representaciones locales y globales

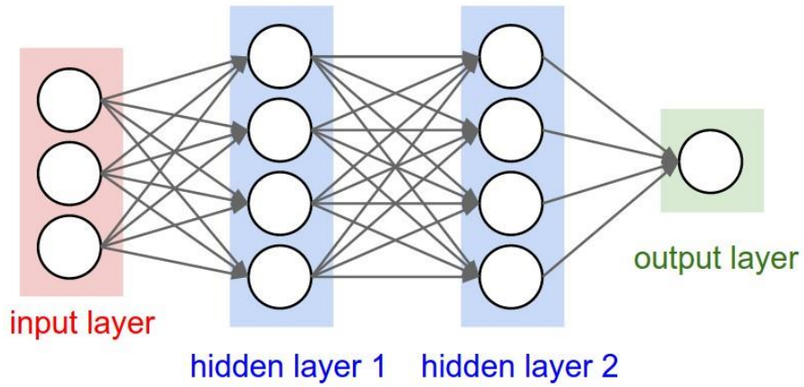


Buscamos un mecanismo de aprendizaje suficientemente poderoso y general, que permita aprender representaciones que capturen aspectos complejos y eventualmente desconocidos de los datos, con el fin de aprender a realizar tareas de manera óptima



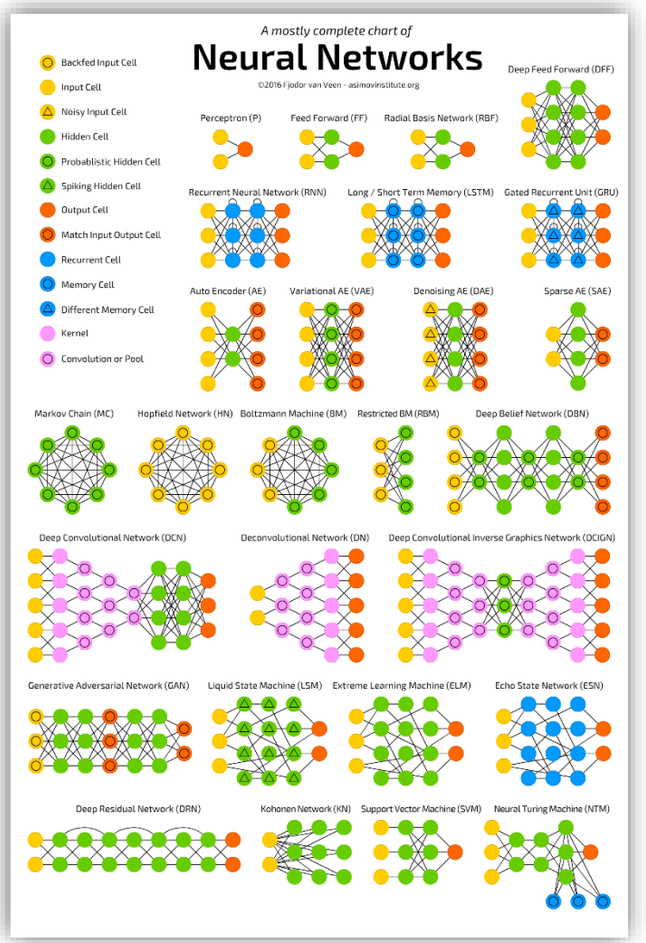
¿Qué son las redes neuronales profundas?
¿Qué es *deep learning*?

- Modelos que aprendan capas de presentaciones composicionales.
- Capas son formadas por unidades (funciones parametrizadas) diferenciables, entrenadas en conjunto (*differentiable programming*) en base a una función objetivo común.
- Al menos un nivel de representaciones intermedias, pero generalmente son muchas.
- Actualmente es el mecanismo de aprendizaje más exitoso, manteniendo récords en gran cantidad de tareas complejas, como reconocimiento de voz, caracteres, objetos, etc.



Redes neuronales son capaces de aprender distintas capas de representaciones

- Método altamente práctico y general para aprender funciones continuas y discretas.
- Brillan en la presencia de grandes volúmenes de datos.
- Existen redes para prácticamente cualquier problema.
- Difíciles de entrenar, interpretabilidad es compleja (caja negra) y sufren de serios problemas de sobreajuste (esto último ya no es tan claro).



A mostly complete chart of

Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

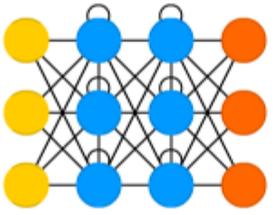
Perceptron (P)

Feed Forward (FF)

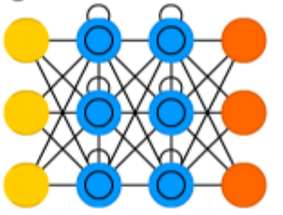
Radial Basis Network (RBF)

Deep Feed Forward (DFF)

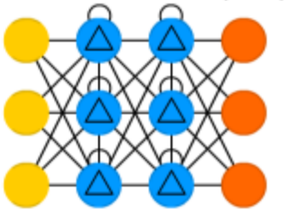
Recurrent Neural Network (RNN)



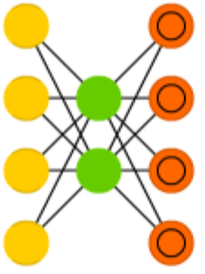
Long / Short Term Memory (LSTM)



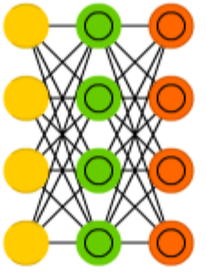
Gated Recurrent Unit (GRU)



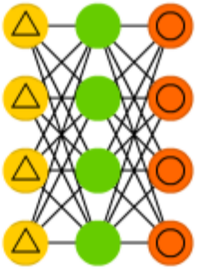
Auto Encoder (AE)



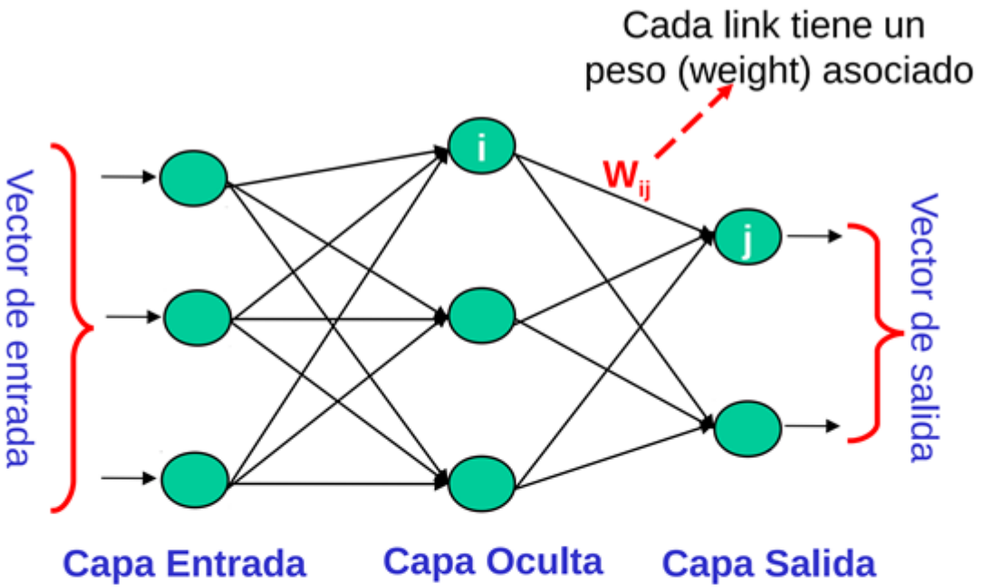
Variational AE (VAE)



Denoising AE (DAE)

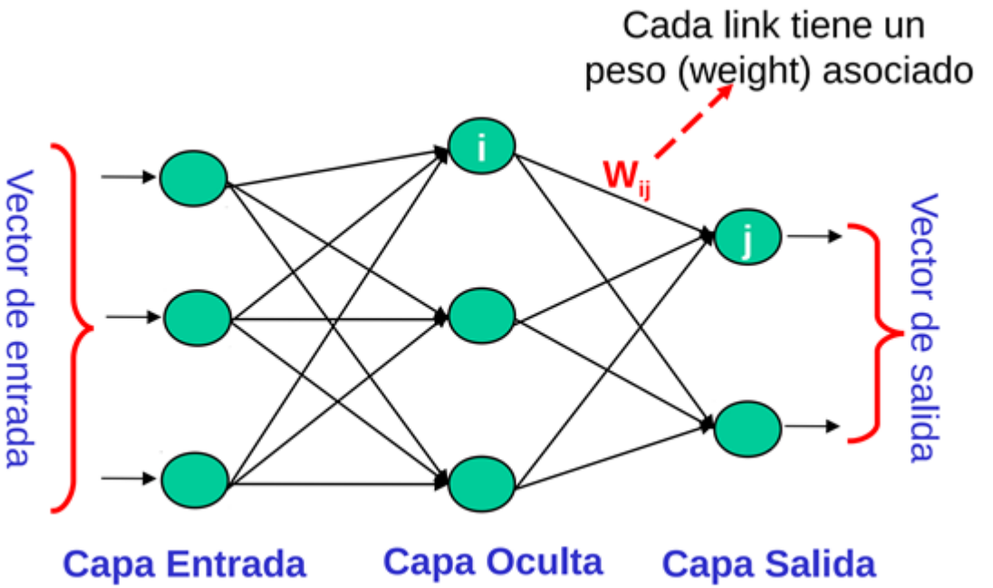


Sparse AE (SAE)



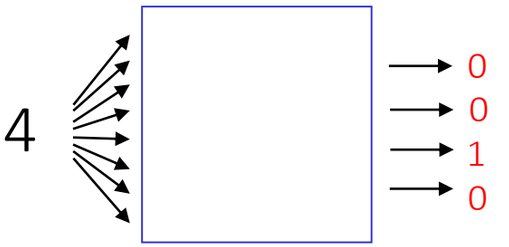
Una FFN es caracterizada por:

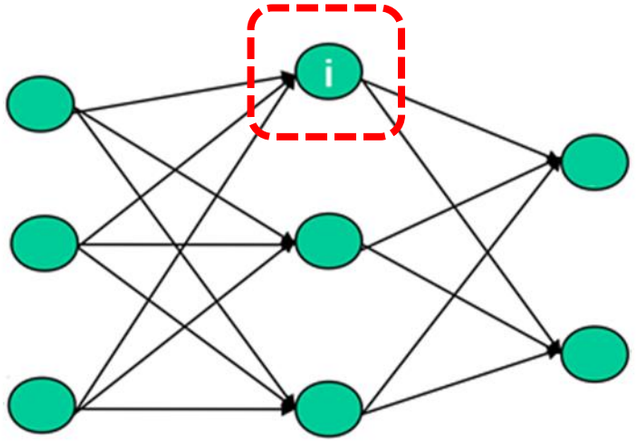
1. Estructura de la red, i.e., número de unidades (neuronas en cada) en cada capa y número de capas ocultas.
2. Método para determinar el valor de los pesos W_{ij} que conectan a una neurona i con una neurona j.
3. Función de activación que modula el impulso que enváa una neurona a otra.



Dada la estructura de la red, ¿cómo podemos ajustar los pesos para que al ingresar un vector de entrada determinado obtengamos la salida deseada?

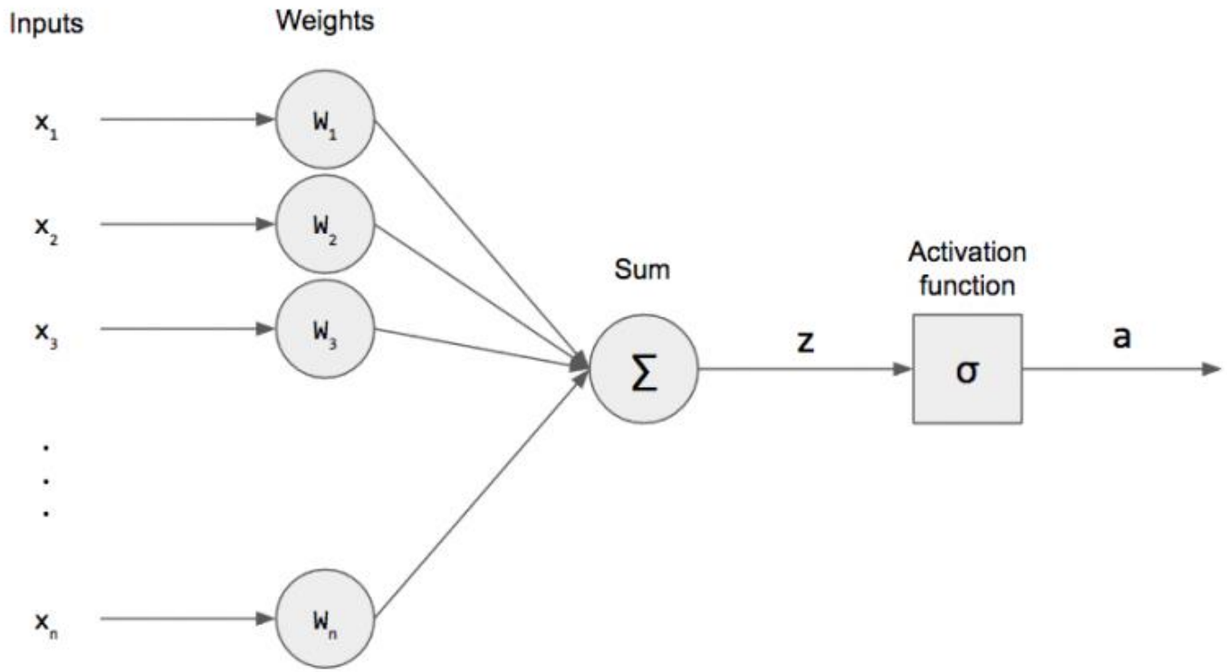
Ejemplo: cuando la red recibe una imagen de un 4, su codificación de salida debe indicar un 4.

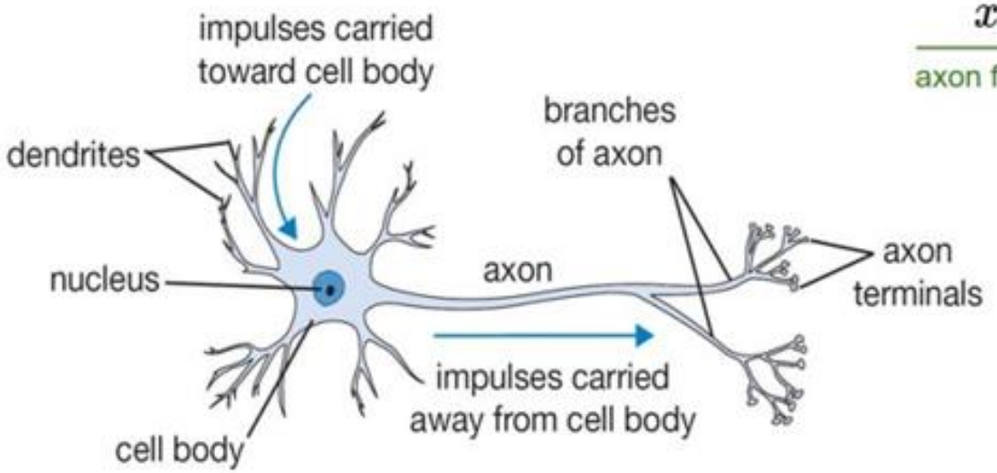


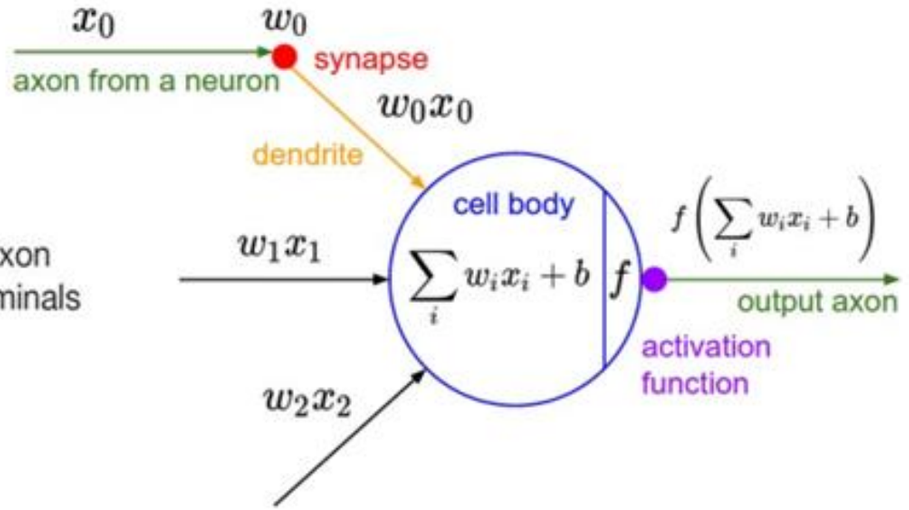

i

1958 Perceptron

La unidad fundamental de una red neuronal se conoce como Perceptron




impulses carried
toward cell body
dendrites
nucleus
cell body
branches
of axon
axon
axon
terminals
impulses carried
away from cell body


x_0
w_0
axon from a neuron
synapse
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation
function
$w_2 x_2$

Un Perceptron permite estimar funciones de manera supervisada

- Si tenemos suficientes datos, pares $(x^{(i)}, y^{(i)})$, es posible construir una función que nos indique cuán buena es en promedio la estimación.
- Definimos entonces una posible función de pérdida (o costo, o error) de la siguiente manera:

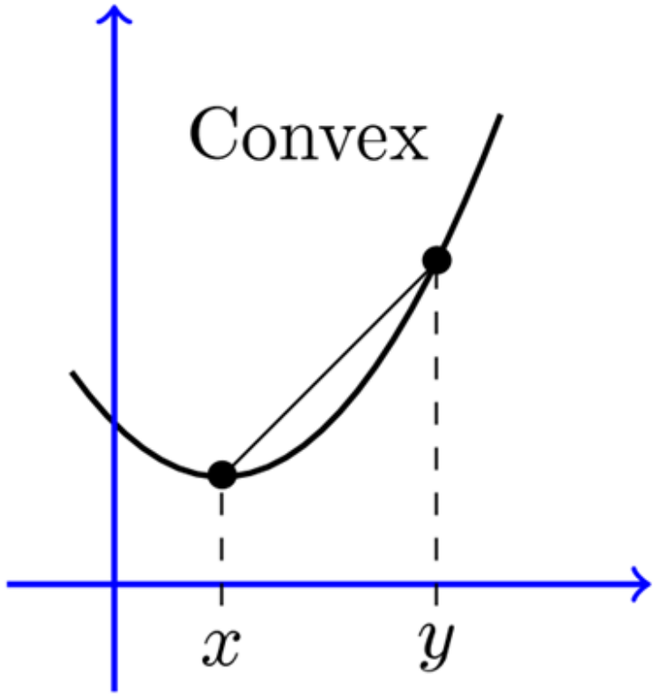
$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- El valor para los pesos que minimice esta pérdida, entregará la mejor estimación de la función original.

¿Cómo podemos entrenar un Perceptron (lineal) para estimar funciones?

1. ¿Qué características tiene la función objetivo elegida?

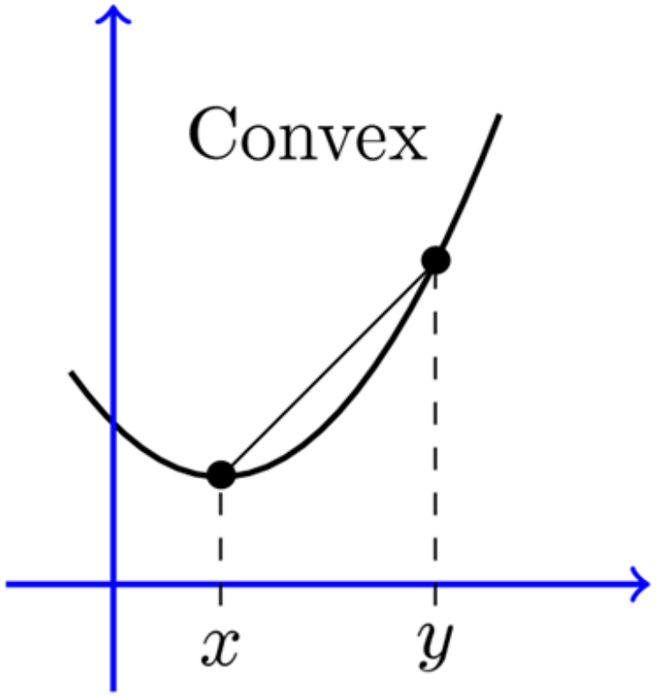
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

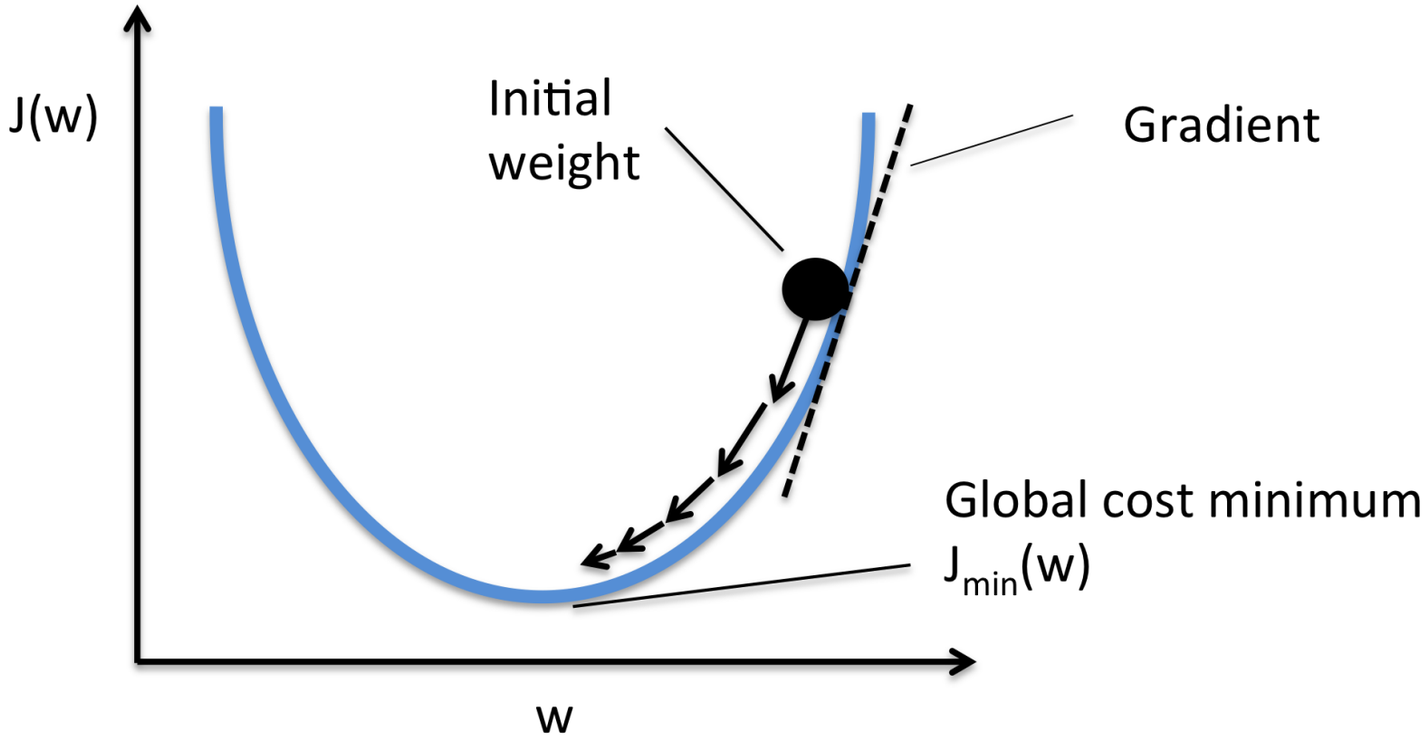


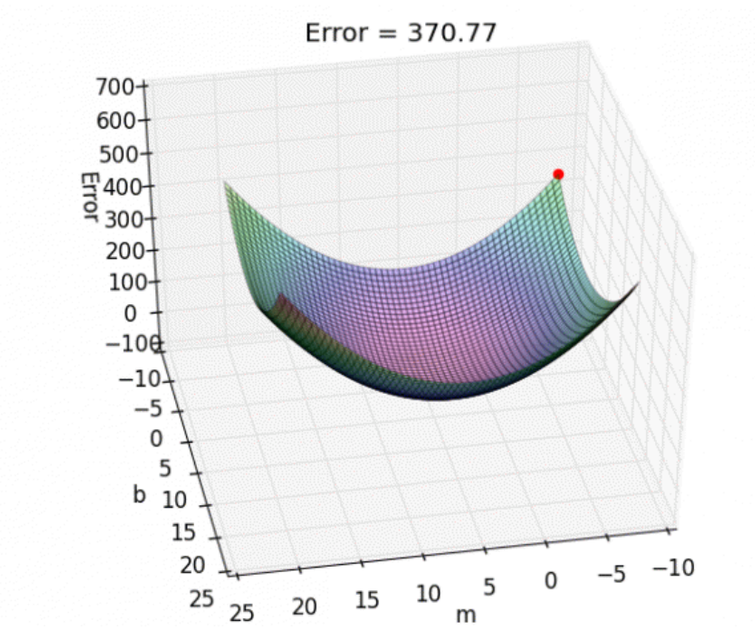
¿Cómo podemos entrenar un Perceptron (lineal) para estimar funciones?

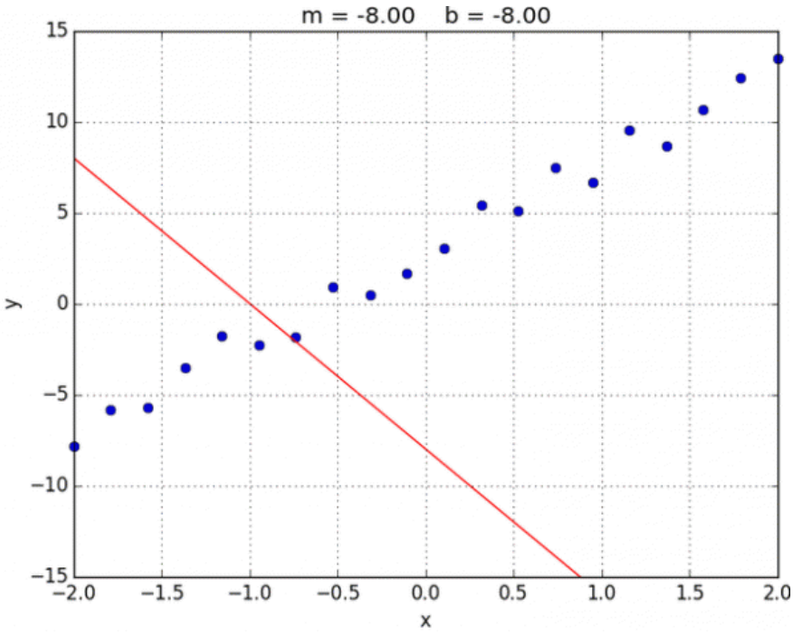
1. ¿Qué características tiene la función objetivo elegida?

2. ¿Qué algoritmo de optimización podemos utilizar para este problema?

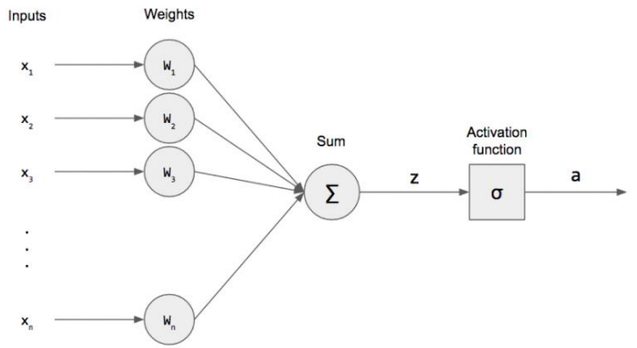



J(w)
Initial
weight
Gradient
Global cost minimum
$J_{min}(w)$
w


Error = 370.77
Error
700
600
500
400
300
200
100
0
−100
b
−10
−5
0
5
10
15
20
25
m
25
20
15
10
5
0
−5
−10

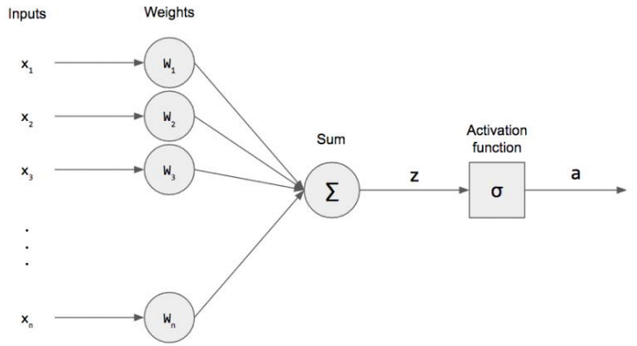

m = -8.00 b = -8.00
y
15
10
5
0
−5
−10
−15
x
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Dado esto, cómo quedaría el problema de optimización para entrenar un Perceptron lineal



$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

¿Y la derivada de la función objetivo?



$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Dado que el problema de aprendizaje es convexo, podemos usar “tranquilamente” descenso de gradiente

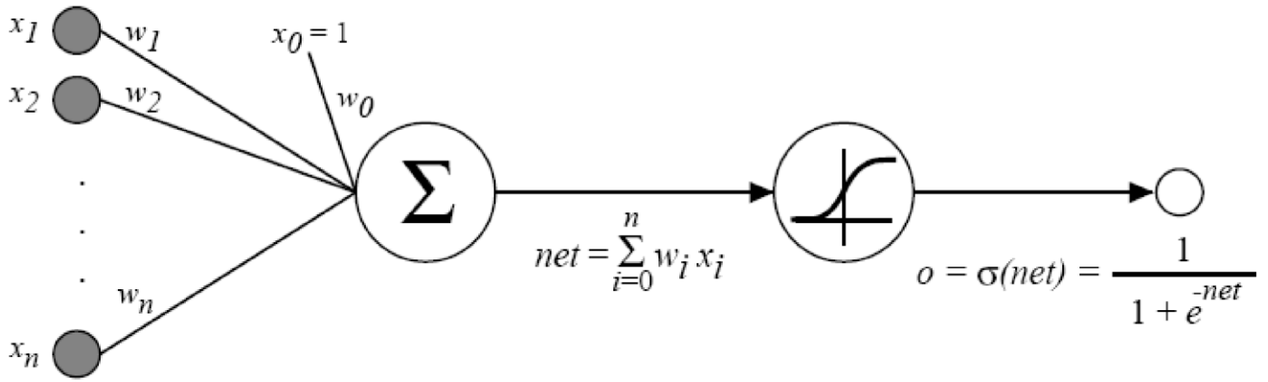
- Inicializar cada peso w_i a un valor aleatorio pequeño (ej. rango [0,1])
- **do**
  - Inicializar gradientes Δw_i a cero
  - **for** (x,y)en set de entrenamiento
    - Aplique (x,y)a perceptron y obtenga salida **out**
    - Para cada peso calcule gradiente y guarde en acumulador:

      $\Delta w_i \leftarrow \Delta w_i + \eta(y - \textbf{out})\, x_i$

    end for
  - Actualice valor de cada peso:

    $w_i^{new} = w_i^{old} + \Delta w_i$
- **while** (condición de término ! = TRUE).

¿Cómo quedaría entonces la regla de actualización para un Perceptrón sigmoidal?



Regla de entrenamiento

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

$$\frac{\partial E}{\partial w_i} = -\sum_{d \in D} (t_d - o_d) o_d (1 - o_d) x_{i,d}$$

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Sistemas Urbanos Inteligentes

Introducción a Redes Neuronales

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación