

Clase 2 - Redes Neuronales

Plan de la presentación

Modelo Neuronal

Aprendizaje

Asociador Lineal

BackPropagation

Entrenamiento y Descriptores

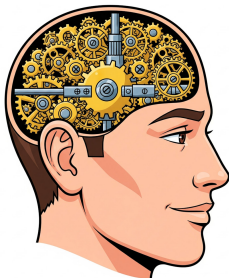
Descriptores

Modelo Neuronal

Modelo de neurona

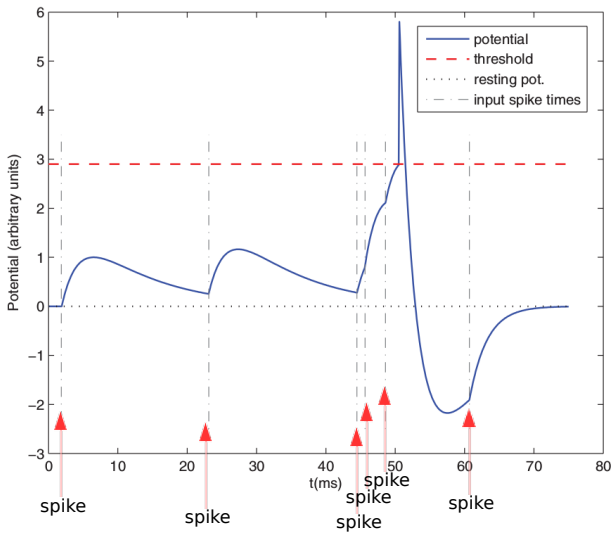
Modelo Biológico

- ▶ Se estima que en nuestro cerebro cohabitan unas cien mil millones de neuronas (10^{11} neuronas).
- ▶ Cada neurona tiene aproximadamente diez mil conexiones con otras neuronas (10^4).
- ▶ A través de estas interconexiones, una neurona recoge señales procedentes de otras neuronas, luego procesa estas señales y transmite otra señal hacia otras neuronas.
- ▶ De esta manera la información se transmite de unas neuronas a otras.



Modelo de neurona

Modelo Biológico



Modelo de neurona

Modelo Biológico

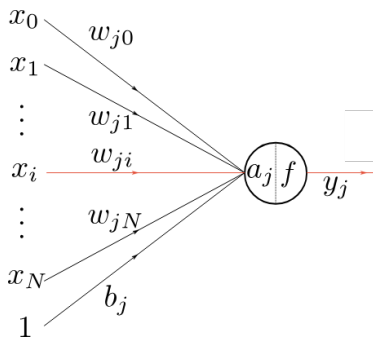
El Modelo Neuronal ofrece diversas ventajas, entre las cuales se destacan:

- ▶ **Aprendizaje Adaptativo:** son capaces de aprender con base en un entrenamiento o experiencia inicial.
- ▶ **Generalización:** una vez entrenada, a la red se le pueden presentar datos distintos a los usados durante el aprendizaje. La respuesta obtenida dependerá del parecido de los datos con los ejemplos de entrenamiento.
- ▶ **Abstracción o tolerancia al ruido:** son capaces de extraer o abstraer las características esenciales de las entradas aprendidas, de esta manera pueden procesar correctamente datos incompletos o distorsionados.
- ▶ **Procesamiento paralelo:** la información es procesada por las neuronas artificiales en forma paralela.
- ▶ **Memoria distribuida:** el conocimiento acumulado por la red se halla distribuido en numerosas conexiones.
- ▶ **Tolerancia a fallos:** una red neuronal es capaz de seguir funcionando adecuadamente a pesar de sufrir lesiones como la destrucción de neuronas o conexiones, ya que la información se halla distribuida por toda la red.

Modelo de neurona

Modelo Matemático

El Modelo Neuronal matemático (perceptrón) de una neurona j

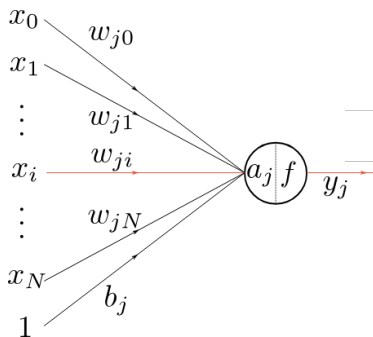


- El set de entradas $\{x_0, \dots, x_N\}$ se conecta a la neurona via

Modelo de neurona

Modelo Matemático

El Modelo Neuronal matemático (perceptrón) de una neurona j

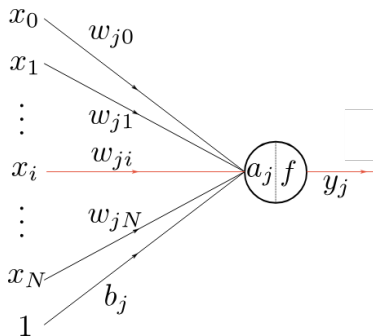


- El set de entradas $\{x_0, \dots, x_N\}$ se conecta a la neurona via
- las sinapsis con sus los pesos $\{\omega_{j0}, \dots, \omega_{jN}\}$ y el bias b_j

Modelo de neurona

Modelo Matemático

El Modelo Neuronal matemático (perceptrón) de una neurona j

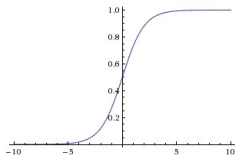


- ▶ El set de entradas $\{x_0, \dots, x_N\}$ se conecta a la neurona via
- ▶ las sinapsis con sus los pesos $\{\omega_{j0}, \dots, \omega_{jN}\}$ y el bias b_j
- ▶ acumulando la excitación en $a_j = b_j + \sum_i \omega_{ji} x_i$

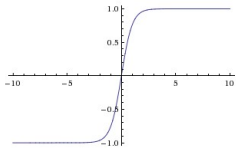
Modelo de neurona

Modelo Matemático

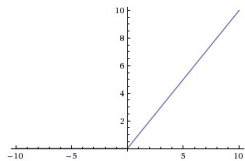
Las funciones de activación f más utilizadas hoy en día son:



sigmoide



tanh



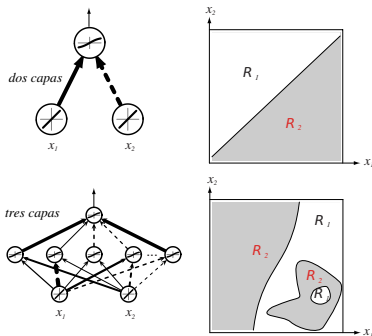
relu

Lo importante es que estas funciones deben ser *derivables*, para luego ser utilizadas en el aprendizaje. La selección de cual función de activación utilizar depende de la aplicación.

Modelo de neurona

Modelo Matemático: Arquitectura

- La arquitectura de base del perceptrón posee dos capas y permite resolver problemas LINEALMENTE SEPARABLES.
- Para problemas LINEALMENTE **NO** SEPARABLES, la solución es agregar capas a la red, incorporando no linealidades en la resolución.



Modelo de neurona

Modelo Matemático

- ▶ Las redes multicapa no lineales, compuestas por d unidades de entrada, n_H capas escondidas (*hidden layers*), y c unidades de salida, poseen un enorme poder computacional o expressive power.
- ▶ En clasificación la señal discriminante k , con $k = 1, \dots, c$, de cada una de esas clases es:

$$g_k(\mathbf{x}) = z_k = f \left(\sum_{j=1}^{n_H} \omega_{kj} f \left(\sum_{i=1}^d \omega_{ji} x_i + b_j \right) + b_k \right)$$

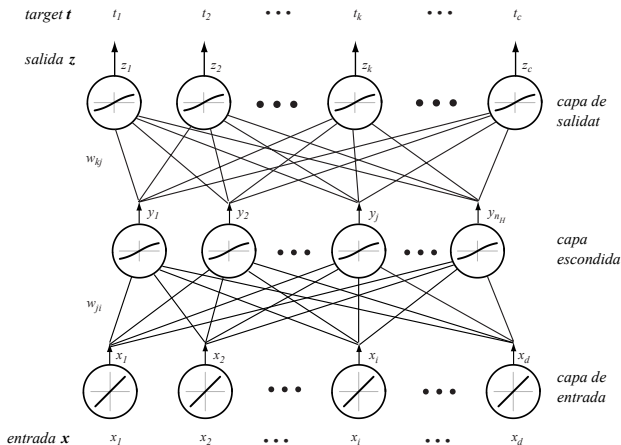
- ▶ Luego, se le asigna la clase C a la instancia de test \mathbf{x} de acuerdo a:

$$C = \operatorname{argmax}_{k=1, \dots, c} g_k(\mathbf{x})$$

- ▶ En teoría, con la ecuación anterior, y dado un suficiente número de neuronas escondidas n_H , se podría representar cualquier función continua de entrada.

Modelo de neurona

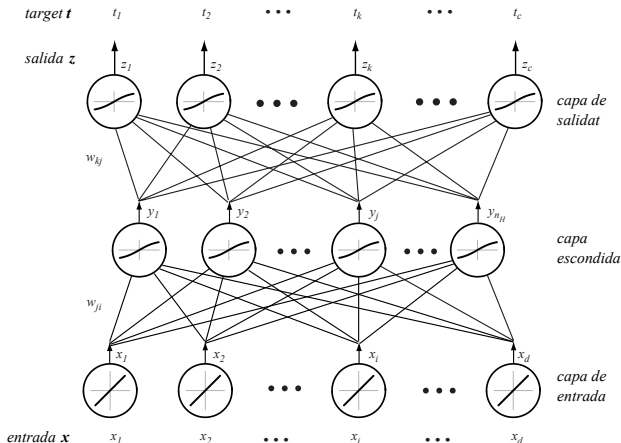
Modelo Matemático



- El problema es que no sabemos de antemano cual es la arquitectura óptima para obtener esto.

Modelo de neurona

Modelo Matemático



- El problema es que no sabemos de antemano cual es la arquitectura óptima para obtener esto.
- Y TENEMOS QUE ENTRENAR LA RED!!!

Aprendizaje

Representación formal

Definimos nuestro espacio de datos como los pares:

$$(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^p, y^p) \in \mathcal{X}$$

- ▶ \mathcal{X} set no vacío, también llamado el *dominio*.
- ▶ \mathbf{x}^i patrones, casos, entradas, observaciones, instancias, features.
- ▶ y^i labels, etiquetas, targets, salidas, observaciones.

El objetivo de la clasificación es ser capaz de generar una respuesta robusta ante datos no vistos anteriormente y poder predecir su clase correcta. Teniendo por ejemplo, el par (\mathbf{x}^k, y^k) , durante el entrenamiento se busca una función:

$$f(\mathbf{x}^k) : \mathcal{X} \implies \hat{y}^k$$

que además minimice una función de costo:

$$\min ||y^k - \hat{y}^k||$$

Análisis del dominio

El dominio \mathcal{X} definido por los pares (\mathbf{x}^i, y^i) :

- ▶ Features (\mathbf{x}):
 - ▶ Vectores o matrices en \mathbb{R} .
 - ▶ Conviene que estén centrados y normalizados $\{\pm 1\}$.
- ▶ Targets (y):
 - ▶ 1 elemento y clasificación binaria: los pares $(0, 1)$ o $(-1, 1)$ definen a que clase pertenece la entrada.
 - ▶ 1 elemento y regresión: conviene normalizar los valores.
 - ▶ multiples elementos y clasificación multiclase: representación *one-hot*.

Aprendizaje

Generación de datasets para el aprendizaje

Una de las principales tareas a la hora de enfrentar problemas de clasificación, es la confección de una base de datos para el aprendizaje.

La base debería tener las siguientes características:

- ▶ Lo más grande posible.
- ▶ Alta variabilidad intra-clase.
- ▶ Balanceada en número de ejemplos inter-clases.

Generación de datasets para el aprendizaje

Una de las principales tareas a la hora de enfrentar problemas de clasificación, es la confección de una base de datos para el aprendizaje.

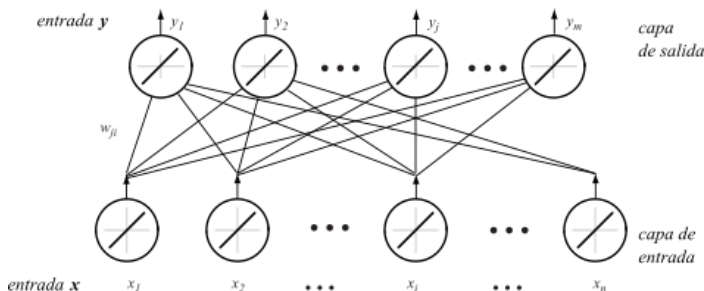
La base debería tener las siguientes características:

- ▶ Lo más grande posible.
- ▶ Alta variabilidad intra-clase.
- ▶ Balanceada en número de ejemplos inter-clases.

Pero todo depende de la problemática a resolver, que estas características se cumplan.

Asociador Lineal

- ▶ El Asociador Lineal es una RNA unidireccional y hace parte de los modelos de redes neuronales estándar.
- ▶ Consta de dos capas: una de entradas denotada por $\mathbf{x} \in \mathbb{R}^n$ y una de salida denotada por $\mathbf{y} \in \mathbb{R}^m$.
- ▶ definimos $W = \{w_{ji}\} \in \mathbb{R}^{m \times n}$ a la matriz de pesos sinápticos donde cada fila de W contiene los pesos presinápticos que llegan a una neurona j .



Aprendizaje

Asociador Lineal

La operación efectuada por el Asociador Lineal se corresponde a la suma ponderada que se aplica luego a la función de activación de tipo identidad. Esto queda:

$$\mathbf{y} = W\mathbf{x}$$

o bien,

$$y_j = \sum_{i=1}^n w_{ji} x_i \quad \text{con } j = 1, \dots, m$$

Asociador Lineal

La operación efectuada por el Asociador Lineal se corresponde a la suma ponderada que se aplica luego a la función de activación de tipo identidad. Esto queda:

$$\mathbf{y} = W\mathbf{x}$$

o bien,

$$y_j = \sum_{i=1}^n w_{ji} x_i \quad \text{con } j = 1, \dots, m$$

El Asociador Lineal vincula los p pares:

$$(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^\mu, y^\mu), \dots, (\mathbf{x}^p, y^p)$$

Se busca entonces la matriz de pesos W , tal que ante entradas similares a x^μ responda con salidas similares a y^μ . Para ello, la regla de aprendizaje utiliza los patrones de entrada con sus respectivas salidas deseadas para establecer el conjunto óptimo de pesos W .

Aprendizaje

Regla de Hebb

En 1949 Donald Hebb establece uno de los mecanismos clásicos del aprendizaje al postular que

Cuando un axón presináptico causa la activación de cierta neurona, la eficacia de la sinapsis que las relaciona se refuerza

Se denomina aprendizaje Hebbiano a un aprendizaje que involucra una modificación en los pesos, w_{ji} , proporcional al producto de una entrada x_i por la salida y_j de la neurona. Es decir:

$$\Delta w_{ji} = \epsilon x_i y_j,$$

Utilizando esta ecuación de corrección a partir de un par, se establece la regla de actualización del Asociador Lineal como:

$$w_{ji}^{new} = w_{ji}^{old} + \Delta w_{ji}$$

Regla de Hebb

La regla de Hebb para el caso del Asociador Lineal se puede formalizar con la siguiente dinámica de adaptación:

1. En $t = 0$, todos los pesos valen $\{w_{ji} = 0, \forall i = 1, \dots, n, \forall j = 1, \dots, m\}$
2. Para $t = 1, \dots, p$, el par μ es presentado a la red y los pesos son actualizados de acuerdo a la regla de Hebb:

$$w_{ji}^t = w_{ji}^{t-1} + x_i^\mu y_j^\mu \quad \text{con} \quad j = 1, \dots, m \quad i = 1, \dots, n$$

3. Cuando todos los pares son presentados, la fase de adaptación termina resultando en

$$w_{ji} = \sum_{\mu=1}^p x_i^\mu y_j^\mu \quad \text{con} \quad j = 1, \dots, m \quad i = 1, \dots, n$$

Regla de Hebb

La dinámica de adaptación del Asociador Lineal también se puede escribir en forma matricial:

$$W^{(0)} = 0$$
$$W^{(\mu)} = W^{(\mu-1)} + \mathbf{y}^{\mu} \mathbf{x}^{\mu T} \quad \mu = 1, \dots, p$$

Generalizando, se puede escribir como producto de matrices:

$$W = W^{(p)} = \sum_{\mu=1}^p \mathbf{y}^{\mu} \mathbf{x}^{\mu T} = Y X^T \quad (1)$$

donde las columnas de la matriz $X \in \mathbb{R}^{n \times p}$ y las de la matriz $Y \in \mathbb{R}^{m \times p}$ son los vectores de entrada de x^{μ} y los de salida y^{μ} , respectivamente; es decir,

$$X = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}, Y = \begin{pmatrix} y_{11} & \dots & y_{1p} \\ \vdots & \ddots & \vdots \\ y_{m1} & \dots & y_{mp} \end{pmatrix}$$

Regla de Hebb

Asumamos que los vectores $\{\mathbf{x}^1, \dots, \mathbf{x}^p\}$ es una base ortonormal en \mathbb{R}^n (con lo cual $p \leq n$). Bajo esta condición y utilizando la ecuación 1, el Asociador Lineal responde con la salida \mathbf{y}^μ ante la entrada \mathbf{x}^μ cumpliéndose:

$$\begin{aligned} W\mathbf{x}^\mu &= (\mathbf{y}^1\mathbf{x}^{1T} + \dots + \mathbf{y}^p\mathbf{x}^{pT})\mathbf{x}^\mu \\ &= \mathbf{y}^1(\mathbf{x}^{1T}\mathbf{x}^\mu) + \dots + \mathbf{y}^p(\mathbf{x}^{pT}\mathbf{x}^\mu) \\ &= \mathbf{y}^\mu \end{aligned}$$

Regla de Hebb

Asumamos que los vectores $\{\mathbf{x}^1, \dots, \mathbf{x}^p\}$ es una base ortonormal en \mathbb{R}^n (con lo cual $p \leq n$). Bajo esta condición y utilizando la ecuación 1, el Asociador Lineal responde con la salida \mathbf{y}^μ ante la entrada \mathbf{x}^μ cumpliéndose:

$$\begin{aligned} W\mathbf{x}^\mu &= (\mathbf{y}^1\mathbf{x}^{1T} + \dots + \mathbf{y}^p\mathbf{x}^{pT})\mathbf{x}^\mu \\ &= \mathbf{y}^1(\mathbf{x}^{1T}\mathbf{x}^\mu) + \dots + \mathbf{y}^p(\mathbf{x}^{pT}\mathbf{x}^\mu) \\ &= \mathbf{y}^\mu \end{aligned}$$

Si eliminamos la condición de ortogonalidad, pero mantenemos la de norma igual a 1,

$$\begin{aligned} W\mathbf{x}^\mu &= \mathbf{y}^1(\mathbf{x}^{1T}\mathbf{x}^\mu) + \dots + \mathbf{y}^p(\mathbf{x}^{pT}\mathbf{x}^\mu) \\ &= \mathbf{y}^\mu + \sum_{v \neq \mu} \mathbf{y}^v(\mathbf{x}^{vT}\mathbf{x}^\mu) \\ &= \mathbf{y}^\mu + \delta \end{aligned}$$

La expresión anterior se denomina expansión señal-ruido y proporciona la salida deseada mas un término adicional.

Regla de la pseudo-inversa

Si no es posible escoger una matriz de pesos tal que las ecuaciones

$$W\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \quad \forall \mu = 1, \dots, p$$

sean satisfechas en forma **exacta**, interesa entonces encontrar una solución aproximada.

Regla de la pseudo-inversa

Si no es posible escoger una matriz de pesos tal que las ecuaciones

$$W \mathbf{x}^\mu = \mathbf{y}^\mu \quad \forall \mu = 1, \dots, p$$

sean satisfechas en forma **exacta**, interesa entonces encontrar una solución aproximada.

En este sentido, los algoritmos de aprendizaje se deducen de a partir de una función que mida el error a la salida de la red, por ejemplo, el error cuadrático medio:

$$E(W) = \frac{1}{p} \sum_{\mu=1}^p \|\mathbf{y}^\mu - W \mathbf{x}^\mu\|^2 = \frac{1}{p} \sum_{\mu=1}^p \sum_{i=1}^n (y_i^\mu - W_i^T x_i^\mu)^2$$

Regla de la pseudo-inversa

Si no es posible escoger una matriz de pesos tal que las ecuaciones

$$W \mathbf{x}^\mu = \mathbf{y}^\mu \quad \forall \mu = 1, \dots, p$$

sean satisfechas en forma **exacta**, interesa entonces encontrar una solución aproximada.

En este sentido, los algoritmos de aprendizaje se deducen de a partir de una función que mida el error a la salida de la red, por ejemplo, el error cuadrático medio:

$$E(W) = \frac{1}{p} \sum_{\mu=1}^p \|\mathbf{y}^\mu - W \mathbf{x}^\mu\|^2 = \frac{1}{p} \sum_{\mu=1}^p \sum_{i=1}^n (y_i^\mu - W_i^T x_i^\mu)^2$$

El problema de aprendizaje del Asociador Lineal se transforma en el de obtener un conjunto de pesos que minimicen esta expresión.

Regla de la pseudo-inversa

A partir de la norma de Frobenius $\|M\|_F^2$ de una matriz $M \in \mathbb{R}^{p \times q}$ tal que:

$$\|M\|_F^2 = \|(m_{ij})\|_F^2 = \sum_{i=1}^p \sum_{j=1}^q m_{ij}^2$$

Luego, la ecuación de $E(W)$ se puede expresar:

$$E(W) = \frac{1}{p} \|Y - WX\|_F^2$$

Regla de la pseudo-inversa

A partir de la norma de Frobenius $\|M\|_F^2$ de una matriz $M \in \mathbb{R}^{p \times q}$ tal que:

$$\|M\|_F^2 = \|(m_{ij})\|_F^2 = \sum_{i=1}^p \sum_{j=1}^q m_{ij}^2$$

Luego, la ecuación de $E(W)$ se puede expresar:

$$E(W) = \frac{1}{p} \|Y - WX\|_F^2$$

A partir de esta formulación, una regla de aprendizaje basada en la utilización de la pseudo-inversa de Moore-Penrose puede escribirse como:

$$W = YX^+$$

donde X^+ es la pseudo-inversa de X . Esta elección para W minimiza el error cuadrático medio.

- ▶ El Asociador Lineal es una representación muy simple de una red neuronal.
- ▶ Los pesos de las interconexiones entre las neuronas de entrada y salida pueden obtenerse analíticamente.
- ▶ Es de esperar que el método del aprendizaje a partir del error sea más robusto.
- ▶ PROBLEMA: entrenar una arquitectura multicapa se dificulta y fue la razón por la cual las redes neuronales fueron abandonadas durante años.

Backpropagation

- ▶ En 1982 se retoman las redes neuronales a partir del entrenamiento por BackPropagation (1974) y Rumelhart y Hinton lo aplican de forma exitosa en 1986.
- ▶ Backpropagation es uno de los métodos más simples e instructivos para el entrenamiento supervisado de redes neuronales.
- ▶ Vamos a estudiar la matemática aplicada a la arquitectura multicapas.

Backpropagation

Función de costo (o error en el entrenamiento)

$$J(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2 = \frac{1}{2} \|\mathbf{t} - \mathbf{z}\|^2 \quad (2)$$

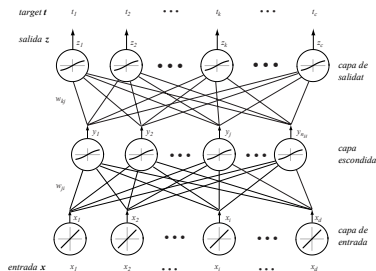
La regla de aprendizaje sigue el descenso del gradiente:

$$\begin{aligned} \Delta \mathbf{w} &= -\eta \frac{\partial J}{\partial \mathbf{w}} \\ \Delta w_{pq} &= -\eta \frac{\partial J}{\partial w_{pq}} \end{aligned} \quad (3)$$

η es la variable de aprendizaje.

La ley de actualización de los pesos en la iteración m es simplemente:

$$\mathbf{w}(m+1) = \mathbf{w}(m) + \Delta \mathbf{w}(m)$$

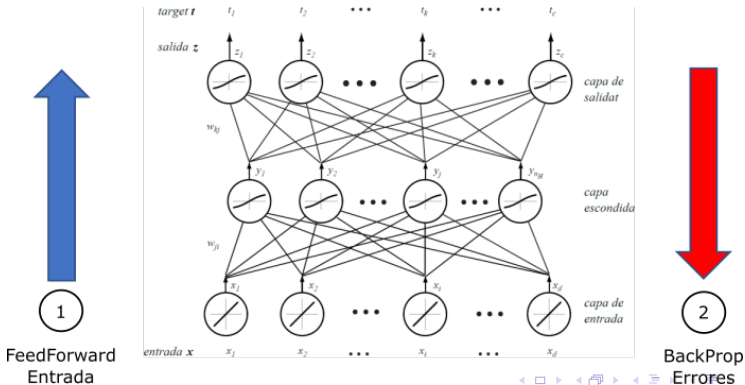


Backpropagation

Dos Etapas

El Backpropagation se aplica en dos etapas

1. FeedForward: donde se propaga una entrada a la red hasta la última capa. Se reservan todos los valores de activación de las células de cada capa, y las derivadas (gradiente) de cada una de ellas.
2. BackProp: se propaga el error calculado por la función de costo cometido por la última capa, a través de la red hasta la primera capa. Esto me permite obtener los updates de las variables de la red.



Backpropagation

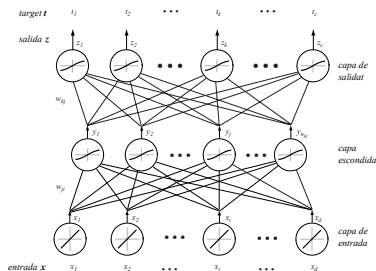
El cálculo del gradiente de las sinapses entre la entrada a la capa escondida es mas sutil.

- De la ecuación 3:

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \quad (6)$$

- El primer término involucra todos los pesos w_{kj}

$$\begin{aligned} \frac{\partial J}{\partial y_j} &= \frac{\partial}{\partial y_j} \left[\frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2 \right] \\ &= - \sum_{k=1}^c (t_k - z_k) \frac{\partial z_k}{\partial y_j} \\ &= - \sum_{k=1}^c (t_k - z_k) \frac{\partial z_k}{\partial a_k} \frac{\partial a_k}{\partial y_j} \\ &= - \sum_{k=1}^c (t_k - z_k) f'(a_k) w_{kj} \end{aligned}$$



Backpropagation

El cálculo del gradiente de las sinapses entre la entrada a la capa escondida es mas sutil.

- El segundo término de la ecuación 6 sale de

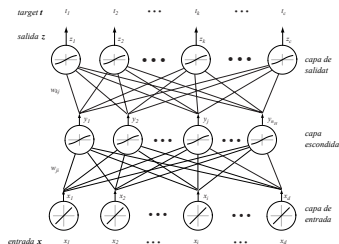
$$\frac{\partial y_j}{\partial a_j} = f'(a_j)$$

- El tercer término de la ecuación 6 se deduce de:

$$a_j = \sum_{i=0}^d x_i w_{ji}$$

con lo cual queda

$$\frac{\partial a_j}{\partial w_{ji}} = x_i$$



Backpropagation

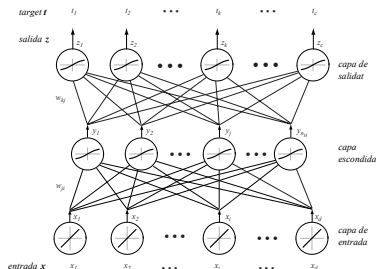
El cálculo del gradiente de las sinapses entre la entrada a la capa escondida es mas sutil.

- ▶ La sensibilidad de una neurona escondida es:

$$\delta_j = f'(a_j) \sum_{k=1}^c w_{kj} \delta_k \quad (7)$$

- ▶ La eq. 7 muestra como el error de una neurona escondida es la suma de las sensibilidades individuales a la neurona de salida ponderada por w_{kj} y multiplicados por $f'(a_k)$.
- ▶ La regla de aprendizaje es:

$$\Delta w_{ji} = \eta x_i \delta_j = \eta \left[\sum_{k=1}^c w_{kj} \delta_k \right] f'(a_j) x_i \quad (8)$$



Backpropagation

Resumen

► Entrada-Escondida

$$\Delta w_{ji} = \eta x_i \delta_j = \eta \left[\sum_{k=1}^c w_{kj} \delta_k \right] f'(a_j) x_i$$

► Escondida-Salida

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(a_k) y_j$$

Backpropagation

Resumen

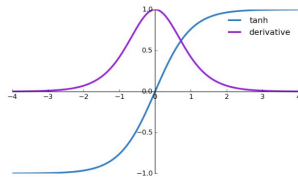
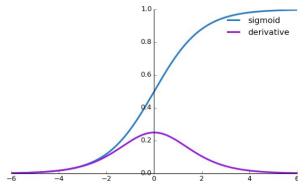
► Entrada-Escondida

$$\Delta w_{ji} = \eta x_i \delta_j = \eta \left[\sum_{k=1}^c w_{kj} \delta_k \right] f'(a_j) x_i$$

► Escondida-Salida






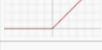



$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(a_k) y_j$$

Importancia de las derivadas



Backpropagation

Importancia de las derivadas

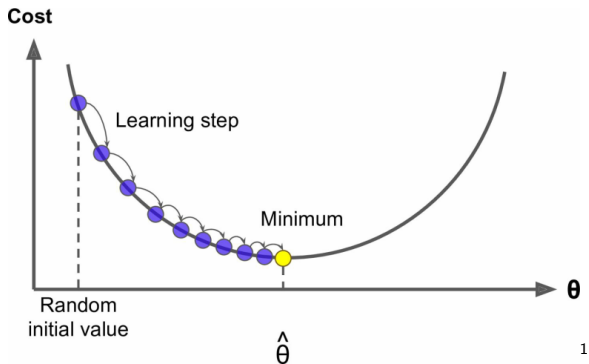
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Entrenamiento y Descriptores

Entrenamiento

Convergencia de la función de costo

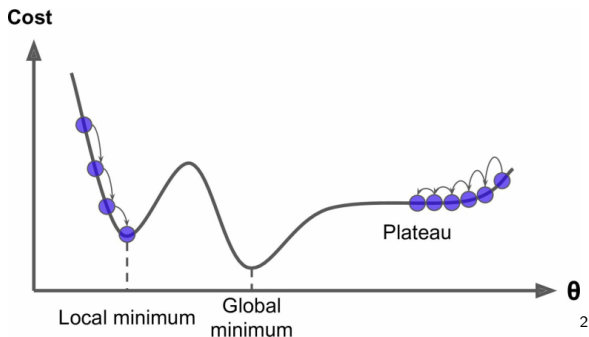
Retomando la función de costo, por ejemplo, a partir del RMSE, se busca que el descenso del gradiente defina los parámetros de la red para encontrar el mínimo de la función convexa.



1

Convergencia de la función de costo

Retomando la función de costo, por ejemplo, a partir del RMSE, se busca que el descenso del gradiente defina los parámetros de la red para encontrar el mínimo de la función convexa.



2

Entrenamiento

Generación de datasets para el entrenamiento

La capacidad resolutive de un modelo de machine learning es la precisión que obtiene frente a ejemplos del dominio que nunca había visto.

Por ello, comunmente se separa la base de entrenamiento en:

- **Base de Entrenamiento:**

$$E = (\mathbf{x}_1^e, y_1^e), \dots, (\mathbf{x}_m^e, y_m^e) \in \mathcal{X}$$

- **Base de Validación:**

$$V = (\mathbf{x}_1^v, y_1^v), \dots, (\mathbf{x}_n^v, y_n^v) \in \mathcal{X}$$

y además:

$$\forall (\mathbf{x}_i^e, y_i^e) \in E \implies \neg \exists (\mathbf{x}_k^v, y_k^v) \in V \wedge (\mathbf{x}_i^e, y_i^e) = (\mathbf{x}_k^v, y_k^v)$$

Entrenamiento

Generación de datasets para el entrenamiento

La capacidad resolutive de un modelo de machine learning es la precisión que obtiene frente a ejemplos del dominio que nunca había visto.

Por ello, comunmente se separa la base de entrenamiento en:

- **Base de Entrenamiento:**

$$E = (\mathbf{x}_1^e, y_1^e), \dots, (\mathbf{x}_m^e, y_m^e) \in \mathcal{X}$$

- **Base de Validación:**

$$V = (\mathbf{x}_1^v, y_1^v), \dots, (\mathbf{x}_n^v, y_n^v) \in \mathcal{X}$$

y además:

$$\forall (\mathbf{x}_i^e, y_i^e) \in E \implies \neg \exists (\mathbf{x}_k^v, y_k^v) \in V \wedge (\mathbf{x}_i^e, y_i^e) = (\mathbf{x}_k^v, y_k^v)$$

Son Independientes!!

Entrenamiento

Generación de datasets para el entrenamiento

La dinámica de entrenamiento usando la base de entrenamiento y validación consiste en:

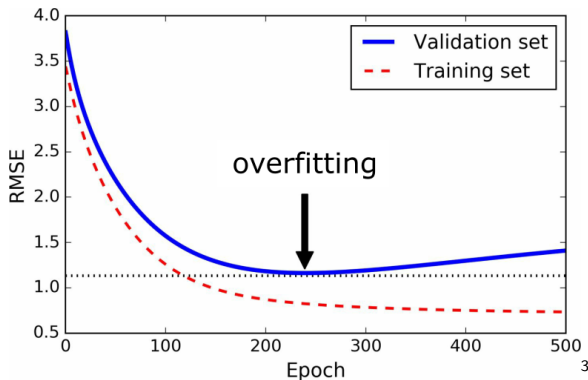
- ▶ Utilizo la base de entrenamiento E para ajustar los pesos y bias de la red neuronal.
- ▶ Evalúo la performance del modelo en la base de validación.

Comunmente se divide el dataset total en un 70 – 30.

Entrenamiento

Generación de datasets para el entrenamiento

Durante el entrenamiento, la función de *loss* calculada en la base de entrenamiento decrece monótonamente. Por su parte, la función de *loss* calculada sobre la base de validación puede caer en *overfitting*.



Entrenamiento

Generación de datasets para el entrenamiento

Contar con datos acotados siempre es un limitante a la hora de encarar un entrenamiento/validación. Una forma donde *todos* los datos se usaron tanto para entrenamiento y validación es el *k-fold cross-validation*.

K-FOLD CROSS VALIDATION

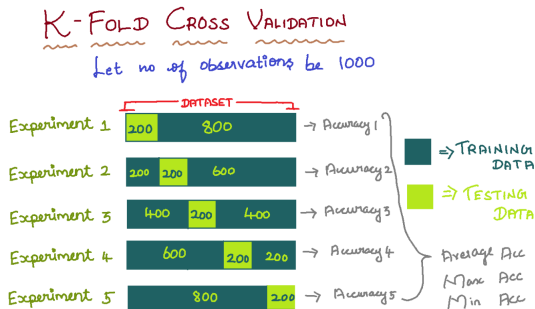
Let no of observations be 1000



Entrenamiento

Generación de datasets para el entrenamiento

Contar con datos acotados siempre es un limitante a la hora de encarar un entrenamiento/validación. Una forma donde *todos* los datos se usaron tanto para entrenamiento y validación es el *k-fold cross-validation*.



4

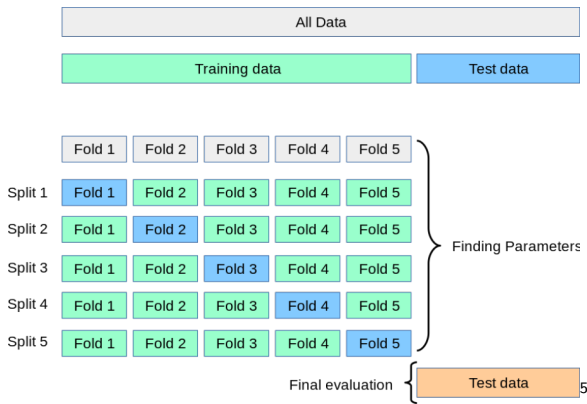
De contar con **muy** pocos datos, se puede usar leave-one-out approach, que lleva el cross-validation al extremo con $K = N$, donde N es la cantidad de ejemplos de entrenamiento.

⁴<https://medium.com/nerd-for-tech/cross-validation-and-types-a7498a68f413>

Entrenamiento

Selección de hiperparámetros

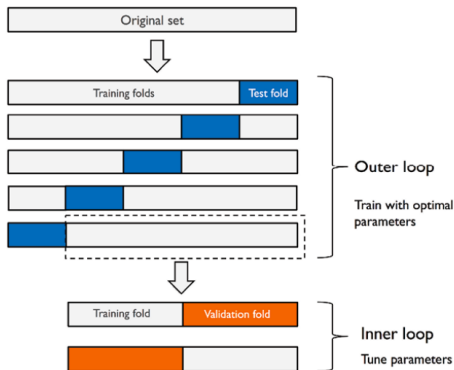
El *k-fold cross-validation* nos permite la selección de los hiperparámetros óptimos del clasificador. El score final debe calcular sobre un tercer dataset totalmente independiente.



⁵https://scikit-learn.org/stable/modules/cross_validation.html

Selección del modelo

El *k-fold cross-validation* nos permite la selección de los hiperparámetros óptimos del clasificador. El score final debe calcular sobre un tercer dataset totalmente independiente.



Descriptores

Descriptores: HOG

Vamos a implementar una red neuronal que clasifique dígitos de matrículas de vehículos multi-estilo usando los descriptores HOG⁷ como espacio de representación.

Vamos a ver antes de continuar, una rápida descripción de HOG.

⁷Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893).

HOG

Los Histogramas de Gradiente Orientado son un descriptor local muy robusto y muy popular en Object Detection: existen implementaciones en muchas librerías de visión artificial.

Entre sus ventajas podemos incluir:

- ▶ Representación compacta de las relaciones de vecindario
- ▶ Cálculo local de orientaciones con overlapping
- ▶ Robusto ante cambio de estilo o contraste

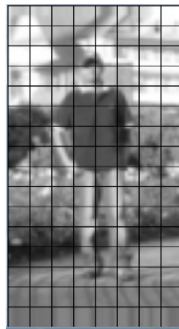
Descriptores: HOG

HOG

En primer lugar se realiza una corrección gamma de la intensidad de la imagen. Luego se divide el parche en una grilla pre-definida.



Corrección
gamma y división
en Celdas



Descriptores: HOG

HOG

Se obtiene el gradiente de la imagen con un operador. Dalal & Triggs terminaron usando el centrado, que permite un cálculo optimizado.

-1	0	1
----	---	---

centrado

-1	1
----	---

simple

1	-8	0	8	-1
---	----	---	---	----

corrección cubica



0	1
-1	0

diagonal

-1	0	1
-2	0	2
-1	0	1

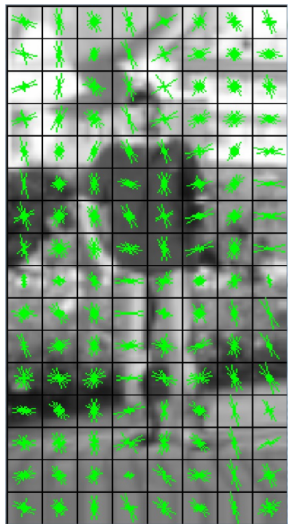
Sobel

HOG

1. Una vez que tengo el gradiente, cuantifico las orientaciones en N valores.
2. Los mejores resultados en el paper se obtienen con $N = 9$.
3. Construyo un histograma para cada celda usando la orientación cuantificada y el módulo del gradiente.
4. El bin b del histograma, son N o sea uno para cada orientación, toma el valor de la suma del gradiente de los píxeles que tienen orientación igual a b .

Descriptores: HOG

HOG



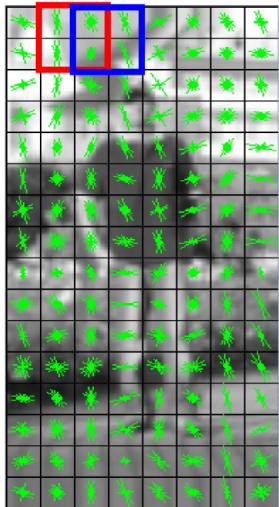
HOG

1. Se definen bloques, que consisten en agrupar las celdas continuas.
2. Los bloques se solapan con el vecino, lo que significa que comparten la información de celdas.
3. Por ejemplo, un bloque puede estar compuesto por 2×2 celdas.

Descriptores: HOG

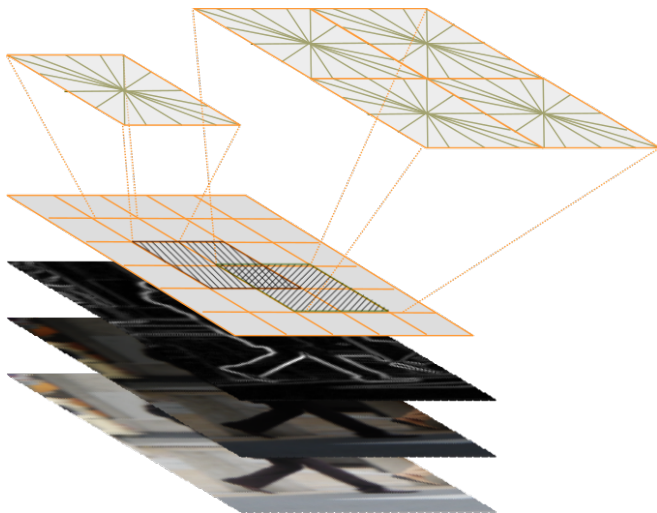
HOG

Bloque A Bloque B



Descriptors: HOG

HOG

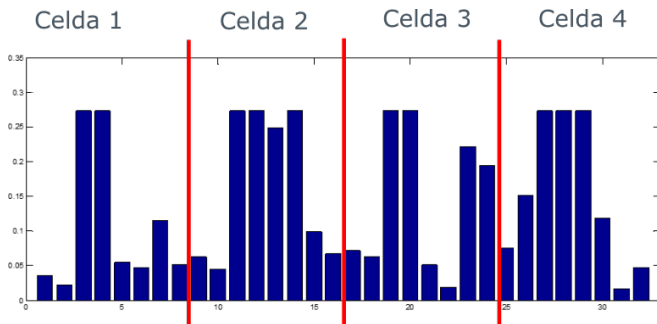


Descriptores: HOG

HOG

Los Histogramas de Gradiente Orientado corresponden a los histogramas de cada celda concatenados de acuerdo al bloque al que pertenecen. Una vez concatenados los valores del histograma del bloque se realiza una normalización $L_2 - norm$.

En el ejemplo, están las 4 celdas del bloque de la cabeza con un valor de $N = 8$.



Conclusiones

Muchas gracias !!!

`pnegri@dc.uba.ar`