Redes Neuronales

Métodos Avanzados de Procesamiento y Síntesis de Imágenes

4to bimestre 2024

Plan de la presentación

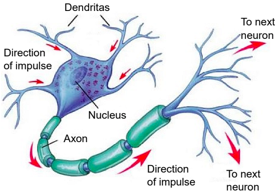
Modelo Neuronal

Modelo de neurona

Modelo Biológico

El Modelo Biológico Neuronal Clásico está compuesto por:

- El cuerpo de la neurona consiste en su núcleo y su membrana
- Las dendritas que son las terminales de entrada al núcleo,
- Un axon es la terminal de salida.
- Las interfaces entre el axon y las dendritas se denominan sinápsis.

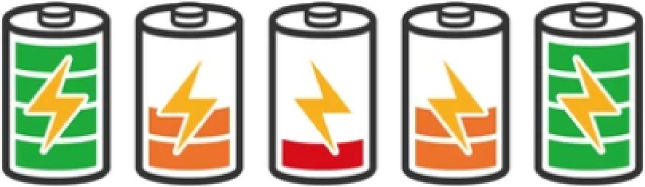


Modelo de neurona

Modelo Biológico

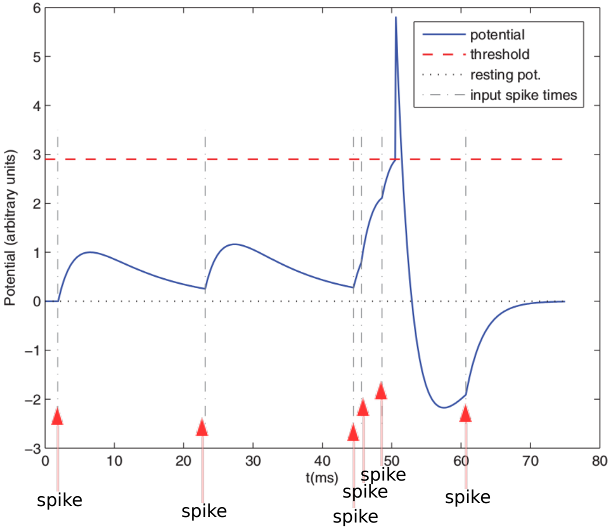
El Modelo Neuronal Clásico funciona de la siguiente manera:

- La dinámica neuronal está controlada por señales eléctricas.
- Estas señales de corta duración se denominan *spikes*.
- Los spikes llegan al núcleo a través de las dendritas y se acumulan en su membrana.
- Cuando el potencial de la membrana alcanza un umbral, la neurona (pre-sináptica) dispara una señal eléctrica a la próxima neurona (post-sináptica) via los axones.



Modelo de neurona

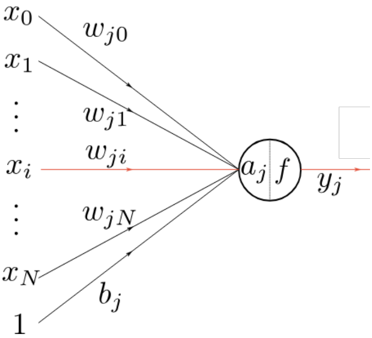
Modelo Biológico



Modelo de neurona

Modelo Matemático

El Modelo Neuronal matemático (perceptrón) de una neurona j

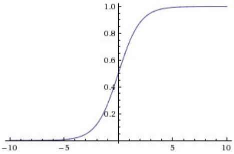


- El set de entradas $\{x_0, ..., x_N\}$ se conecta a la neurona via
- las sinapsis con sus los pesos $\{\omega_{j0}, ..., \omega_{jN}\}$ y el bias b_j
- acumulando la exitación en $a_j = b_j + \sum_i \omega_{ji} x_i$
- que es aplicada a la función de activación f para obtener su salida $y_j = f(a_j)$

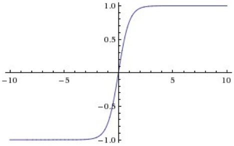
Modelo de neurona

Modelo Matemático

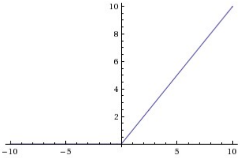
Las funciones de activación f más utilizadas hoy en día son:



sigmoide



tanh



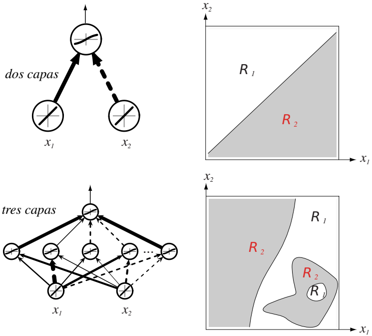
relu

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

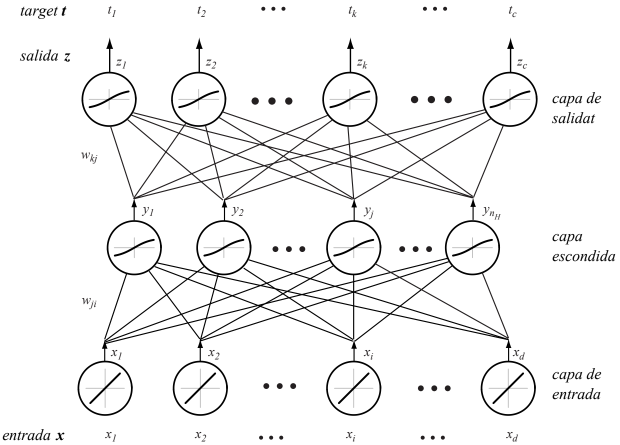
Modelo Matemático

- 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, incorporanto no linealidades en la resolución.



Modelo de neurona

Modelo Matemático



Modelo de neurona

Modelo Matemático

- Las redes multicapa nolineales, 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 $k = 1, ..., c$, de cada una de esas clases es:

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

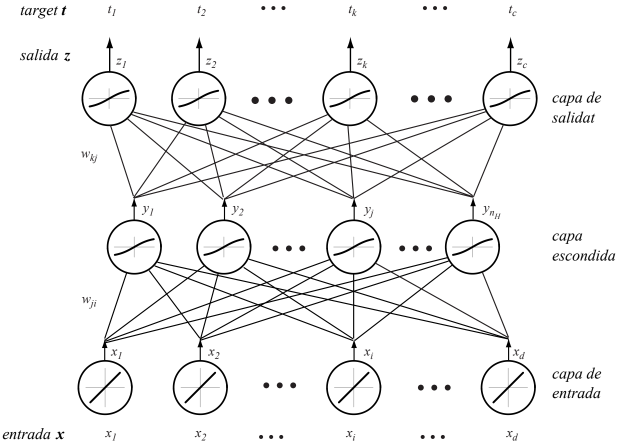
- Luego, se le asigna la clase C a la instancia de test $\mathbf{x}$ de acuerdo a:

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

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

Modelo de neurona

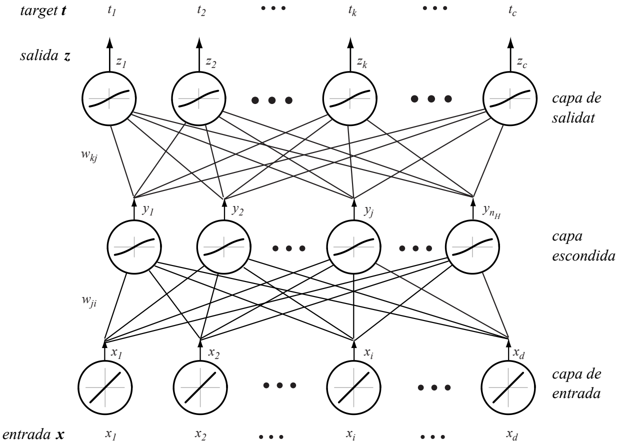
Modelo Matemático



- 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!!!

Backpropagation

Backpropagation

- Backpropagation es uno de los métodos más simples e instructivos para el entrenamiento supervisado de redes neronales multicapas.

Backpropagation

Función de costo (o error en el entrenamiento)

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

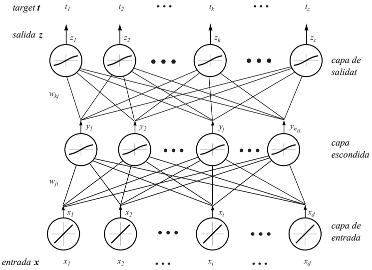
La regla de aprendizaje sigue el descenso del gradiente:

$$\Delta \mathbf{w} = -\eta \frac{\partial J}{\partial \mathbf{w}}$$

$$\Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}} \tag{2}$$

η es la variable de aprendizaje.
La ley de actualización de los pesos en la iteración m es simplemente:

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

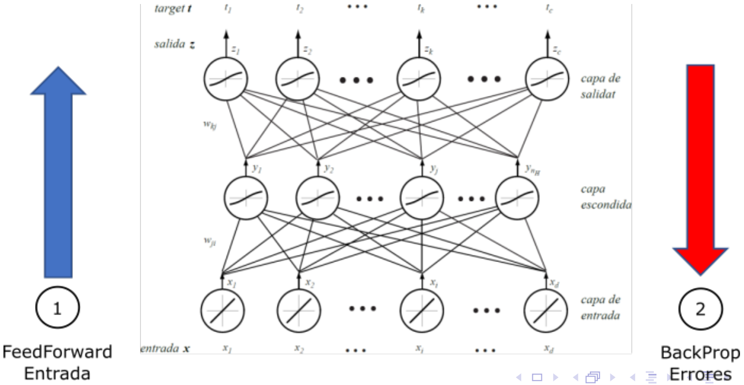


Backpropagation

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 primer capa. Esto me permite obtener los updates de las variables de la red.



Backpropagation

Para resolver la ecuación 2:

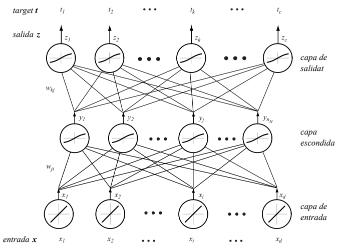
- Considerando los pesos de la capa escondida a la salida w_{kj}, usamos la regla de la cadena:

$$\frac{\partial J}{\partial w_{pq}} = \frac{\partial J}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = -\delta_k \frac{\partial a_k}{\partial w_{kj}} \quad (3)$$

  donde δ_k es la sensibilidad de la neurona de salida k que muestra como cambia el error total a partir de su activación.

- Aplicando la regla de la cadena, derivando eq. 1 y suponiendo que f' existe, obtenemos la expresión de δ_k:

$$\delta_k = -\frac{\partial J}{\partial a_k} = \frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial a_k} = (t_k - z_k) f'(a_k)$$



Backpropagation

- La segunda derivada de eq. 3 se deduce de:

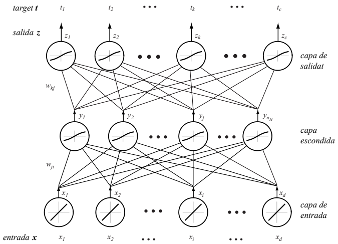
$$a_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj}$$

donde el bias b_k se integra al vector de pesos. Luego:

$$\frac{\partial a_k}{\partial w_{kj}} = y_j$$

- Agrupando, obtenemos el gradiente de los pesos de la capa escondida a la salida:

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(a_k) y_j \tag{4}$$



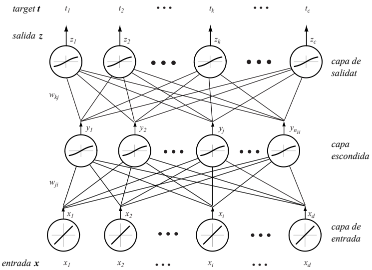
Backpropagation

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

- De la ecuación 2:

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \qquad (5)$$

- 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

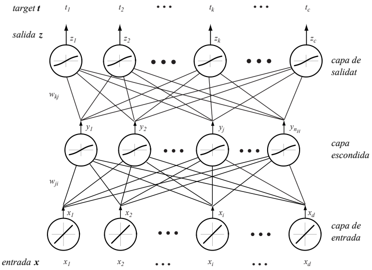
El cálculo del gradiente de las sinapses entre la entrada a la capa escondida es mas sutil.

- La sensibilidad de una neurona escondida es:

$$\delta_j = f'(a_j) \sum_{k=1}^{c} w_{kj}\delta_k \qquad (6)$$

- La eq. 6 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 $f'(a_j)$.
- La regla de aprendizaje es:

$$\Delta w_{ji} = \eta x_i \delta_j = \eta \left[\sum_{k=1}^{c} w_{kj}\delta_k \right] f'(a_j) x_i \qquad (7)$$



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

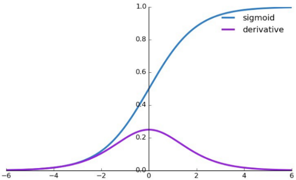
- Entrada-Escondida

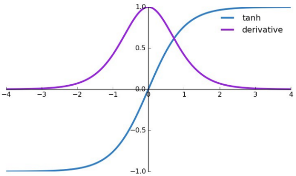
$$\Delta w_{ji} = \eta x_i \delta_j = \eta \left[\sum_{k=1}^{c} w_{kj} \delta_k \right] f'(a_j) x_i$$

- Escondida-Salida

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

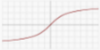
Importancia de las derivadas





Backpropagation

Importancia de las derivadas

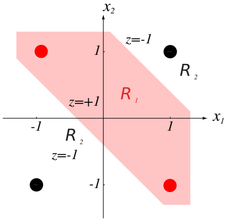
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
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)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric 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		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

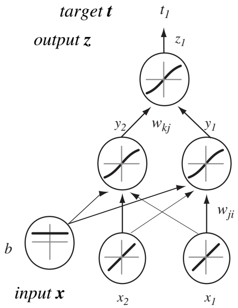
Taller Backprop

EJERCICIO 1

Se va a resolver el problema del XOR utilizando una red con una capa intermedia de 2 neuronas, y una neurona de salida. Las funciones de activación son de tipo **tanh**. Vamos a simplificar las cuentas fijando el valor de $b = 1$. Se considera la siguiente tabla de verdad:

x_1	x_2	z
-1	-1	-1
-1	1	1
1	1	-1
1	-1	1





EJERCICIO 1

Tareas:

- Completar el código del archivo `xor.py`.
- Analizar a través de gráficos, la evolución de los pesos de las neuronas y la sensibilidad en función de la iteración.
- Multiplicar los pesos iniciales de W_h y W_o por 0.1.
- Analizar la convergencia en este caso y plantear nuevamente los gráficos de pesos y sensibilidades.

Aprendizaje

Aprendizaje

Representación formal

Definimos nuestro espacio de datos como los pares:

$$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m) \in \mathcal{X}$$

- $\mathcal{X}$ 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:

$$f(\mathbf{x}_k) : \mathcal{X} \Longrightarrow \hat{y_k}$$

que además minimice una función de costo:

$$\min ||y_k - \hat{y_k}||$$

Aprendizaje

Análisis del dominio

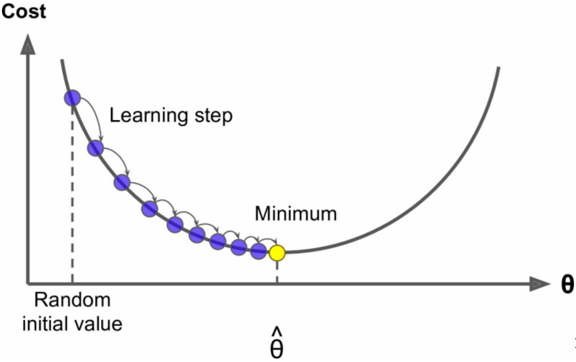
El dominio $\mathcal{X}$ definido por los pares $(\mathbf{x}_i, y_i)$:

- Features $(\mathbf{x})$:
  - Vectores o matrices en $\Re$.
  - Conviene que estén centrados y normalizados $\{\pm 1\}$.
- Targets (y_k):
  - 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

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.



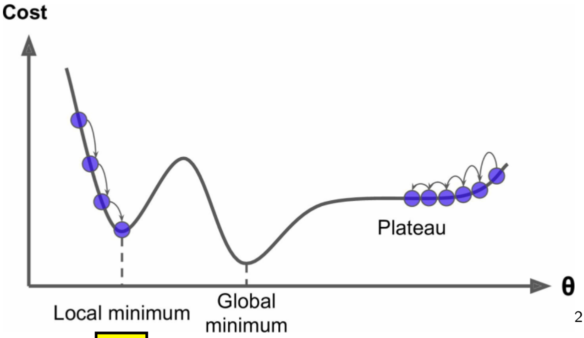
[1]

[1] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.

Aprendizaje

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]

[2] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.

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.

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.

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

Aprendizaje

Generación de datasets para el aprendizaje

La capacidad resolutiva 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 aprendizaje en:

- **Base de Entrenamiento**:

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

- **Base de Validación**:

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

y además:

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

Generación de datasets para el aprendizaje

La capacidad resolutiva 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 aprendizaje en:

- **Base de Entrenamiento**:

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

- **Base de Validación**:

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

y además:

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

Son Independientes!!

Aprendizaje

Generación de datasets para el aprendizaje

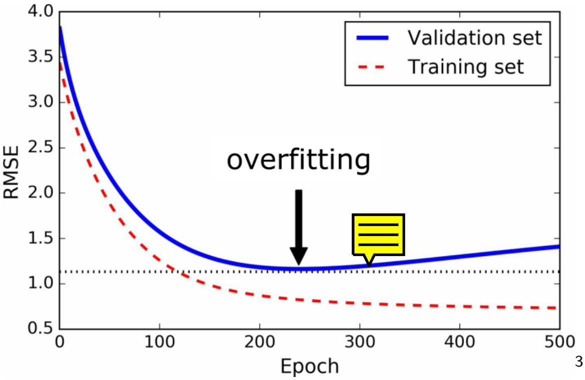
La dinámica de aprendizaje usando la base de entrenamiento y validación consiste en:

- 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 $70 - 30$.

Generación de datasets para el aprendizaje

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*.



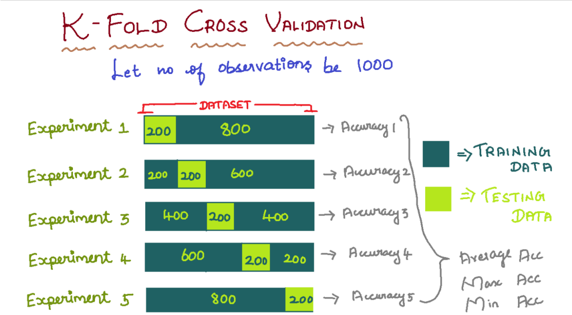
[3]

[3] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.

Aprendizaje

Generación de datasets para el aprendizaje

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*.



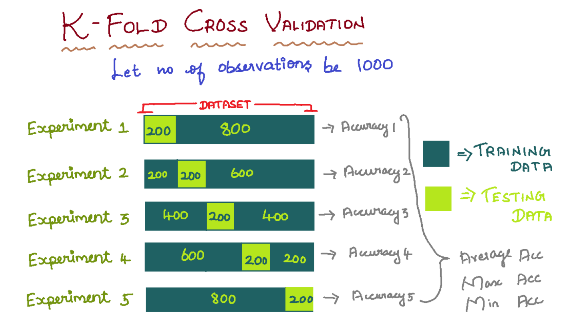
4

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

Aprendizaje

Generación de datasets para el aprendizaje

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]

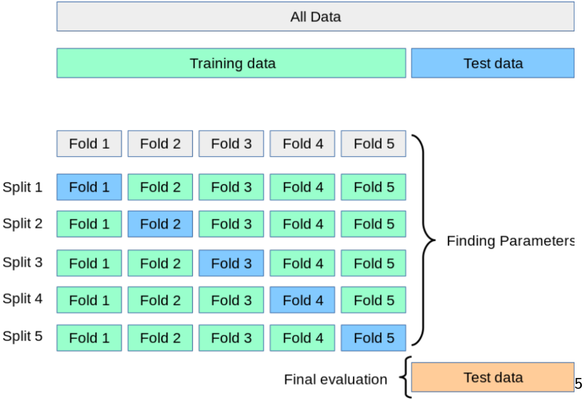
De contar con **muy** pocos datos, se puede usar leave-one-out approach, que lleva el cross-validation al extremo con $K = N$, donde N es la cantidad de ejemplos de aprendizaje.

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

Aprendizaje

Selección de hyperparámetros

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

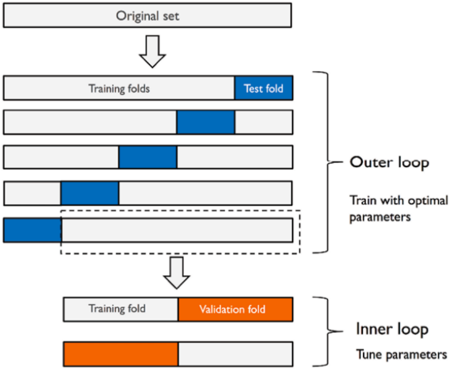


[5]

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

Selección del modelo

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



[6]

[6] https://vitalflux.com/python-nested-cross-validation-algorithm-selection/

Taller Aprendizaje

EJERCICIO 2

Vamos a implementar una red neuronal que clasifique digitos de matrículas de vehículos multi-estilo usando los descriptores HOG[7] como espacio de representación.

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

[7] Dalal, 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).

EJERCICIO 2

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

EJERCICIO 2

HOG

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

EJERCICIO 2

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

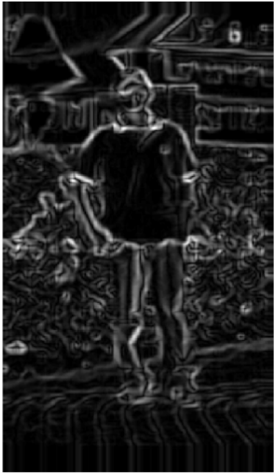
centrado

-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

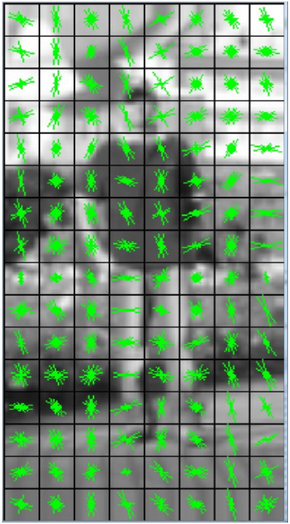
EJERCICIO 2

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.

EJERCICIO 2

HOG



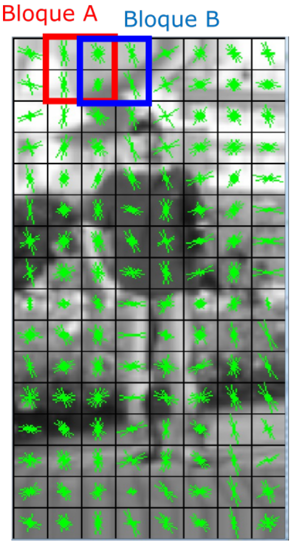
EJERCICIO 2

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 2x2 celdas.

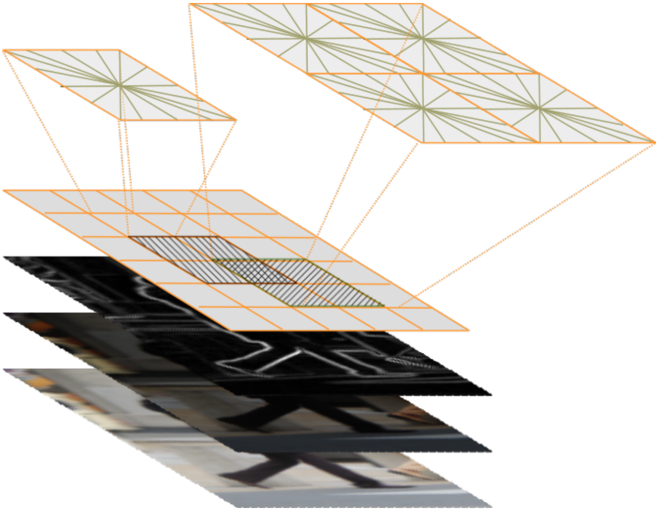
EJERCICIO 2

HOG



EJERCICIO 2

HOG

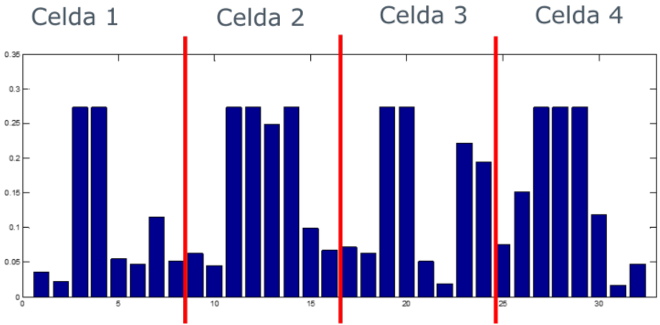


EJERCICIO 2

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 $N = 8$.



EJERCICIO 2

Vamos a implementar una red neuronal que clasifique digitos de matrículas de vehículos multi-estilo usando los descriptores HOG[8] como espacio de representación.

Datos

- Descargar de `http://www.ipol.im/pub/art/2018/173/` el archivo **svmsmo_1.tar.gz** (source code).
- Descomprimir el archivo en un folder temporal.
- Recuperar el folder con las imágenes de los dígitos en: svm_smo/SVMCode/Datasets/BaseOCR_MultiStyle y copiarlo en el folder labo/datos.

[8]Dalal, 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).

EJERCICIO 2

Vamos a implementar una red neuronal que clasifique digitos de matrículas de vehículos multi-estilo usando los descriptores HOG[9] como espacio de representación.

Codigo Pytorch

En el folder $\mathrm{ej2/}$ se encuentra una implementación de una red MLP basada en Pytorch.

Ejercitación

En este laboratorio se pide:

1. Implementar la función de activación *sigmoide* y su derivada.
2. Reemplazar el modelo de MLP de Pytorch por una implementación propia, con las mismas dimensiones pero sin bias. Entrenar este modelo basados en el código del Ej1.
3. Realizar una nueva versión del modelo, pero agregando el bias a las dos capas. Entrenar el modelo con el bias.
4. Agregar una nueva capa escondida del mismo tamaño que la Nh (18 neuronas). Entrenar el modelo completo. Verificar la velocidad de convergencia.

[9] Dalal, 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).

Conclusiones

Muchas gracias !!!

pnegri@dc.uba.ar