Conceptos Fundamentales

Proyecto HAR: CNN-1D + MLP

15 Preguntas Esenciales Respondidas

Con ejemplos numericos reales y graficos explicativos

Contenido:

Semillas aleatorias • Sensores • Frecuencia de muestreo
Estructura de datos • Metricas • Hiperparametros
Funciones de entrenamiento • Calculo de parametros

Índice

1. Por que siempre es necesario fijar una semilla

Reproducibilidad

Una **semilla aleatoria** (random seed) es un numero que inicializa el generador de numeros pseudo-aleatorios. Fijar la semilla garantiza que obtengas **exactamente los mismos resultados** cada vez que ejecutes el codigo.

1.1. Por que los numeros son pseudo-aleatorios

Las computadoras no pueden generar numeros verdaderamente aleatorios. En su lugar, usan **algoritmos deterministas** que producen secuencias que *parecen* aleatorias pero son completamente predecibles si conoces el estado inicial (la semilla).

Generador Congruencial Lineal

$$X_{n+1} = (a \cdot X_n + c) \mod m$$

Donde:

- X_0 = semilla inicial
- a, c, m = constantes del algoritmo

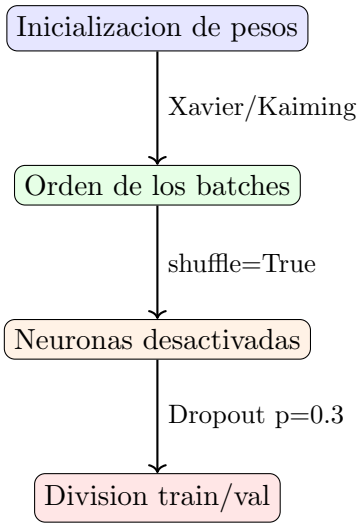
Ejemplo numerico

Con semilla $X_0 = 42$, $a = 1103515245$, $c = 12345$, $m = 2^{31}$:

$$X_1 = (1103515245 \times 42 + 12345) \mod 2^{31} = 1250496027$$
$$X_2 = (1103515245 \times 1250496027 + 12345) \mod 2^{31} = 1116302264$$

Siempre obtendras estos mismos numeros con semilla 42.

1.2. Donde afecta la aleatoriedad en HAR



1.3. Codigo para fijar semillas

```python
import torch
import numpy as np
import random

# Fijar TODAS las fuentes de aleatoriedad
torch.manual_seed(42)          # PyTorch CPU
np.random.seed(42)             # NumPy
random.seed(42)                # Python estandar

# Si usas GPU
torch.cuda.manual_seed(42)     # PyTorch GPU
torch.backends.cudnn.deterministic = True
```

1.4. Consecuencias de NO fijar semilla

Ejecucion	Sin semilla	Con semilla 42
1	94.2 %	95.64 %
2	95.8 %	95.64 %
3	93.1 %	95.64 %
4	96.3 %	95.64 %

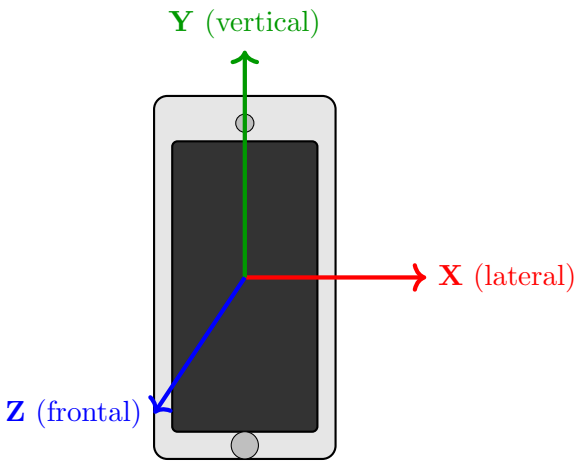
Sin semilla fija no puedes: reproducir resultados, comparar experimentos, depurar errores, ni publicar resultados verificables.

2. Que significa (x, y, z) en cada archivo

Sistema de coordenadas

Los ejes **X, Y, Z** representan las tres dimensiones espaciales del movimiento, medidas desde la perspectiva del telefono. Cada archivo (body_acc_x, body_acc_y, etc.) contiene las mediciones de UN solo eje.

2.1. Orientacion de los ejes



Vista frontal del telefono

2.2. Interpretacion en la vida real

Eje	Movimiento	Ejemplo en actividades
X	Lateral (izquierda-derecha)	Balanceo de caderas al caminar
Y	Vertical (arriba-abajo)	Impacto de cada paso, subir escaleras
Z	Frontal (adelante-atras)	Inclinacion del torso, acostarse

Datos reales del archivo body_acc_x_train.txt

```
1.8085e-04  1.0139e-02  9.2756e-03  5.0659e-03 ...
```

Interpretacion:

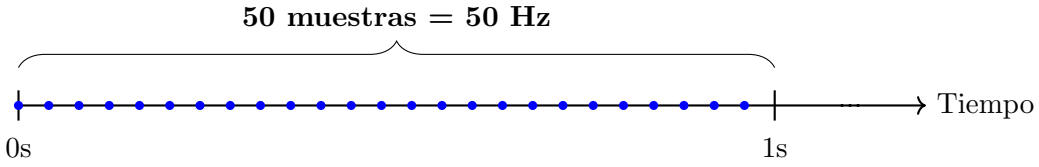
- Cada numero es la aceleracion en el eje X en un instante
- Valores cercanos a 0 = poco movimiento lateral
- Unidades: fracciones de g (gravedad = $9.8\,m/s^2$)

3. Que son 50Hz

Frecuencia de muestreo

50 Hz significa que el sensor toma **50 mediciones por segundo**. Cada "muestra" es una lectura instantanea de la aceleracion o velocidad angular.

3.1. Analogia con fotogramas de video



3.2. Calculo matematico

Relacion Hz - segundos - muestras

$$\text{Frecuencia} = 50 \text{ Hz} = 50 \text{ muestras/segundo}$$

$$\text{Periodo entre muestras} = \frac{1}{50} = 0.02 \text{ segundos} = 20 \text{ ms}$$

$$\text{Muestras en ventana} = 128 \text{ muestras}$$

$$\text{Duracion de ventana} = \frac{128}{50} = 2.56 \text{ segundos}$$

Teorema de Nyquist

Para capturar una senal sin perdida de informacion, la frecuencia de muestreo debe ser **al menos el doble** de la frecuencia maxima de la senal.

$$f_{muestreo} \geq 2 \cdot f_{max}$$

Movimientos humanos: 0-10 Hz. Con 50 Hz podemos capturar hasta 25 Hz.

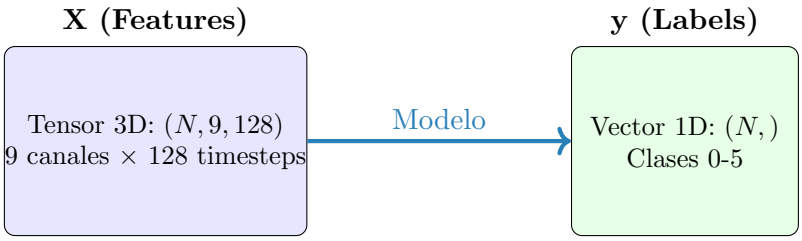
4. Que significa realmente X e Y

Notacion estandar en ML

- **X** (mayuscula) = **Features** o caracteristicas de entrada
- **y** (minuscula) = **Labels** o etiquetas de salida

El objetivo del modelo es aprender la funcion $f : X \rightarrow y$

4.1. En nuestro proyecto HAR



Una muestra individual

X[0] (primera muestra): Shape $(9, 128)$

```
Canal 0 (body_acc_x): [1.81e-04, 1.01e-02, 9.28e-03, ...]
Canal 1 (body_acc_y): [-2.38e-03, -4.12e-03, 1.05e-02, ...]
...
Canal 8 (total_acc_z): [9.81e-01, 9.79e-01, 9.82e-01, ...]
```

y[0] (etiqueta): 5 → LAYING (acostado)

Convencion matematica

- **X mayuscula**: Representa una **matriz** (multiples features)
- **y minuscula**: Representa un **vector** (una dimension)

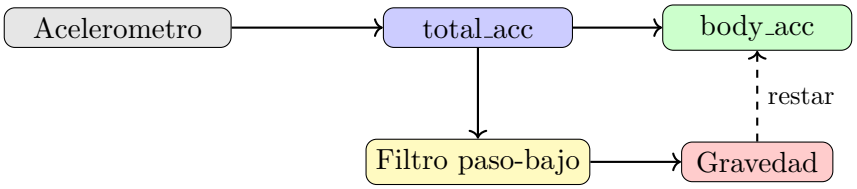
$$\mathbf{X} \in \mathbb{R}^{N \times C \times T}, \quad \mathbf{y} \in \mathbb{Z}^{N}$$

5. Como funcionan body_acc, total_acc y body_gyro

Los tres tipos de senales

Senal	Que mide
total_acc	Aceleracion TOTAL (movimiento + gravedad)
body_acc	Aceleracion del CUERPO (solo movimiento, sin gravedad)
body_gyro	Velocidad ANGULAR (rotacion del dispositivo)

5.1. Diagrama: De donde viene cada senal



Relacion: body_acc = total_acc − gravedad

5.2. Por que necesitamos AMBAS senales

Complementario

Senal	Informacion unica	Ejemplo
total_acc	Orientacion del dispositivo	Detecta si estas acostado o de pie
body_acc	Movimiento puro	Distingue caminar de estar quieto

STANDING vs LAYING

Ambas actividades tienen **body_acc ≈ 0** (sin movimiento).
Pero se distinguen por **donde esta la gravedad**:

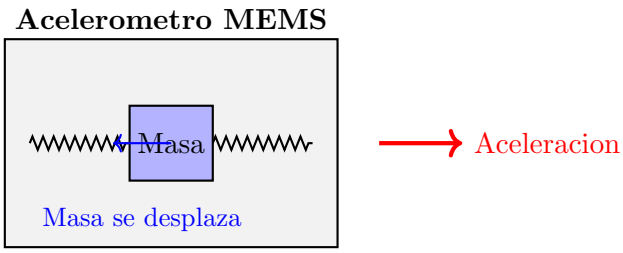
- STANDING: total_acc_y ≈ 1g (persona vertical)
- LAYING: total_acc_z ≈ 1g (persona horizontal)

6. Como funciona un acelerometro y un giroscopio

6.1. El Acelerometro

Principio fisico: Segunda Ley de Newton

Un acelerometro mide **fuerzas** aplicadas a una masa de prueba interna. Segun $F = ma$, si conocemos la masa y medimos la fuerza, podemos calcular la aceleracion.



Como funciona:

1. Cuando el telefono acelera, la masa interna se "queda atras" por inercia
2. Este desplazamiento cambia la capacitancia entre electrodos
3. El circuito mide el cambio y lo convierte a aceleracion

6.2. El Giroscopio

Principio fisico: Efecto Coriolis

Cuando un objeto vibra y el sistema rota, aparece una fuerza perpendicular llamada **fuerza de Coriolis**. El giroscopio mide esta fuerza para determinar la velocidad angular.

$$F_{Coriolis} = 2m(\vec{v} \times \vec{\omega})$$

6.3. Comparacion

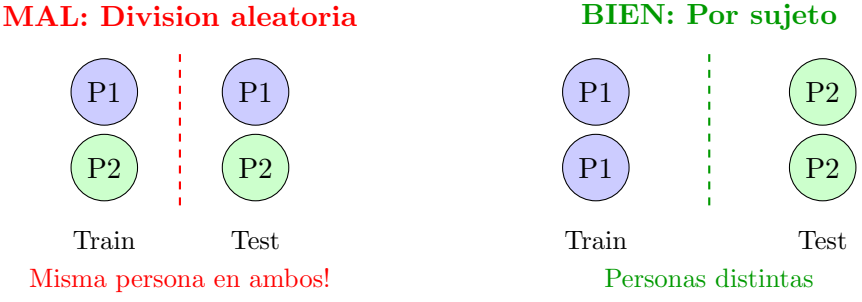
Caracteristica	Acelerometro	Giroscopio
Mide	Aceleracion lineal (m/s^2)	Velocidad angular (rad/s)
Detecta	Movimiento + orientacion	Rotacion
Afectado por gravedad	Si	No
Ejemplo	Paso al caminar	Giro al dar la vuelta

7. Por que las muestras train/test vienen ya divididas

Division por sujetos

El dataset UCI HAR viene pre-dividido porque la division se hizo **por personas**, no aleatoriamente. Esto evita un problema grave llamado **data leakage**.

7.1. El problema del Data Leakage



Memorizacion vs Generalizacion

Si el modelo ve datos de la misma persona en train y test:

- Puede "memorizar" el estilo unico de esa persona
- El accuracy sera **artificialmente alto**
- En produccion, fallara con personas nuevas

7.2. Division del dataset UCI HAR

Conjunto	Sujetos	Muestras
Train	21 personas (70 %)	7,352
Test	9 personas (30 %)	2,947
Total	30 personas	10,299

8. Por que el Accuracy es confiable y otras metricas

8.1. Cuando el Accuracy es confiable

Accuracy en clases balanceadas

El **accuracy** (exactitud) es confiable cuando las clases estan **relativamente balanceadas**.

$$\text{Accuracy} = \frac{\text{Predicciones correctas}}{\text{Total de predicciones}}$$

8.2. Distribucion de nuestras clases

Verificacion de balance

$$\text{Minimo} = 986 \text{ (DOWNSTAIRS)} = 13.4\,\%$$
$$\text{Maximo} = 1407 \text{ (LAYING)} = 19.1\,\%$$
$$\text{Balance perfecto} = 100\,\%/6 = 16.7\,\%$$

Rango 13.4 % - 19.1 % esta cerca del ideal. **Las clases estan balanceadas**.

8.3. El problema con clases desbalanceadas

Deteccion de fraude

Dataset con 99 % legitimas, 1 % fraudulentas.
Un modelo que **siempre predice legitima** tendria:

$$\text{Accuracy} = 99\,\%$$

Pero seria **inutil** porque nunca detecta fraude!

8.4. Otras metricas importantes

Metrica	Pregunta que responde	Formula
Precision	De las predicciones positivas, cuantas son correctas?	$\frac{TP}{TP+FP}$
Recall	De los casos reales positivos, cuantos detectamos?	$\frac{TP}{TP+FN}$
F1-Score	Balance entre Precision y Recall	$2 \cdot \frac{P \cdot R}{P+R}$

Donde: TP = True Positives, FP = False Positives (falsas alarmas), FN = False Negatives (casos perdidos).

9. Que significa Z-score y que es r mayor a 0.5

9.1. Z-score (Estandarizacion)

Z-score

El Z-score transforma datos a una escala estandar con **media 0** y **desviacion estandar 1**. Responde: "A cuantas desviaciones estandar esta este valor de la media?"

$$z = \frac{x - \mu}{\sigma}$$

Ejemplo numerico con datos HAR

Supongamos que para body_acc_x:

- Media $\mu = 0.005$
- Desviacion estandar $\sigma = 0.02$
- Un valor observado $x = 0.045$

$$z = \frac{0.045 - 0.005}{0.02} = \frac{0.04}{0.02} = 2.0$$

Interpretacion: Este valor esta a 2 desviaciones estandar por encima de la media (valor inusualmente alto).

9.2. BatchNorm usa Z-score

En nuestro modelo, **BatchNorm** aplica Z-score dinamicamente:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Donde μ_B y σ_B son la media y desviacion del batch actual.

9.3. Coeficiente de correlacion r

Correlacion de Pearson

El coeficiente r mide la **relacion lineal** entre dos variables. Varia de -1 a +1.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

Interpretacion de valores de r

- $|r| < 0.3$: Correlacion debil o inexistente
- $0.3 < |r| < 0.5$: Correlacion moderada
- $|r| > 0.5$: Correlacion fuerte (relacion notable)
- $|r| > 0.7$: Correlacion muy fuerte

Por eso mostramos correlaciones con $|r| > 0.5$: son las importantes.

10. Que es el Batch y el Batch Size

Definiciones

- **Batch**: Un subconjunto de los datos procesado en una iteracion
- **Batch size**: El numero de muestras en cada batch (nuestro caso: 64)
- **Epoca**: Una pasada completa por todos los datos

10.1. Por que usar batches

Metodo	Ventajas	Desventajas
Full batch	Gradiente exacto	No cabe en memoria
SGD (1 muestra)	Rapido; poca memoria	Gradiente muy ruidoso
Mini-batch (64)	Balance optimo	Requiere elegir tamano

10.2. Calculo numerico

Nuestro proyecto HAR

$$\text{Muestras train} = 7352 \times 0.8 = 5881$$
$$\text{Batch size} = 64$$
$$\text{Batches por epoca} = \lceil 5881/64 \rceil = 92 \text{ batches}$$

11. Ejemplos numericos con formulas explicadas

11.1. Convolucion 1D - Ejemplo paso a paso

Formula de Convolucion 1D

$$(x * w)[t] = \sum_{k=0}^{K-1} x[t+k] \cdot w[k]$$

- x = senal de entrada
- w = kernel (filtro) de tamano K
- t = posicion en la salida

Ejemplo numerico

Entrada: $x = [1, 3, 2, 5, 4]$ (5 timesteps)
Kernel: $w = [1, 0, -1]$ (detector de bordes, $K = 3$)
Calculo (sin padding):

$$\begin{aligned} y[0] &= x[0] \cdot w[0] + x[1] \cdot w[1] + x[2] \cdot w[2] \\ &= 1 \cdot 1 + 3 \cdot 0 + 2 \cdot (-1) = 1 - 2 = \boxed{-1} \end{aligned}$$

$$y[1] = x[1] \cdot 1 + x[2] \cdot 0 + x[3] \cdot (-1) = 3 - 5 = \boxed{-2}$$

$$y[2] = x[2] \cdot 1 + x[3] \cdot 0 + x[4] \cdot (-1) = 2 - 4 = \boxed{-2}$$

Salida: $y = [-1, -2, -2]$ (3 valores $= 5 - 3 + 1$)

11.2. Max Pooling - Ejemplo

Formula de Max Pooling

$$\text{MaxPool}(x, k)[i] = \max_{j=0}^{k-1} x[i \cdot k + j]$$

Toma el maximo de cada ventana de tamano k.

Ejemplo con k=2

Entrada: $x = [3, 1, 4, 1, 5, 9, 2, 6]$ (8 valores)

$$\begin{aligned} y[0] &= \text{máx}(3, 1) = \boxed{3} \\ y[1] &= \text{máx}(4, 1) = \boxed{4} \\ y[2] &= \text{máx}(5, 9) = \boxed{9} \\ y[3] &= \text{máx}(2, 6) = \boxed{6} \end{aligned}$$

Salida: $y = [3, 4, 9, 6]$ (4 valores $= 8/2$)

11.3. Capa Lineal (Fully Connected)

Formula

$$y = W \cdot x + b$$

- $x \in \mathbb{R}^n$ = entrada
- $W \in \mathbb{R}^{m \times n}$ = matriz de pesos
- $b \in \mathbb{R}^m$ = vector de bias
- $y \in \mathbb{R}^m$ = salida

Ejemplo: Linear(3, 2)

Entrada: $x = [1, 2, 3]$
Pesos: $W = \begin{pmatrix} 0.5 & -0.3 & 0.2 \\ 0.1 & 0.4 & -0.2 \end{pmatrix}$, $b = \begin{pmatrix} 0.1 \\ -0.1 \end{pmatrix}$
Calculo:

$$y_0 = (0.5 \times 1) + (-0.3 \times 2) + (0.2 \times 3) + 0.1 = 0.5 - 0.6 + 0.6 + 0.1 = \boxed{0.6}$$
$$y_1 = (0.1 \times 1) + (0.4 \times 2) + (-0.2 \times 3) + (-0.1) = 0.1 + 0.8 - 0.6 - 0.1 = \boxed{0.2}$$

Salida: $y = [0.6, 0.2]$

11.4. ReLU y Softmax

ReLU

$$\text{ReLU}(x) = \text{máx}(0, x)$$

Ejemplo: $\text{ReLU}([-2, 0.5, -0.1, 3]) = [0, 0.5, 0, 3]$

Softmax

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Logits: $x = [2.0, 1.0, 0.1]$
Calculo:

$$e^{2.0} = 7.389, \quad e^{1.0} = 2.718, \quad e^{0.1} = 1.105$$
$$\text{Suma} = 7.389 + 2.718 + 1.105 = 11.212$$
$$P_0 = 7.389/11.212 = \boxed{0.659} \ (66\,\%)$$
$$P_1 = 2.718/11.212 = \boxed{0.242} \ (24\,\%)$$
$$P_2 = 1.105/11.212 = \boxed{0.099} \ (10\,\%)$$

Verificacion: $0.659 + 0.242 + 0.099 = 1.0$ ✓

12. Que es el Test Forward Pass

Forward Pass

El **forward pass** es el proceso de pasar datos a traves de la red neuronal **de entrada a salida**, calculando las activaciones de cada capa.
Test forward pass = hacerlo sin calcular gradientes (solo inferencia).

12.1. Codigo del Test Forward Pass

```
# Test Forward Pass
model.eval()  # Modo evaluacion: desactiva Dropout

with torch.no_grad():  # NO calcular gradientes (ahorra memoria)
    X_batch = X_batch.to(device)  # Mover a GPU
    output = model(X_batch)        # Forward pass

# output.shape = (64, 6) -> 64 muestras, 6 logits

# Convertir logits a probabilidades
probs = F.softmax(output, dim=1)

# Obtener prediccion
predictions = output.argmax(dim=1)  # Indices 0-5
```

12.2. Diferencias Train vs Test

Aspecto	Train	Test
Modo	model.train()	model.eval()
Dropout	Activo	Desactivado
torch.no_grad()	No	Si
Calcula gradientes	Si	No
Uso de memoria	Mayor	Menor

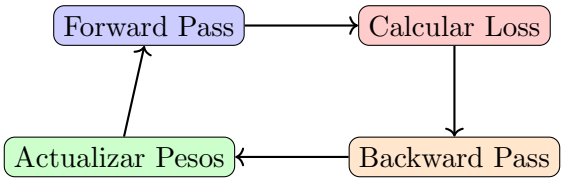
13. Funciones train_epoch, validate_epoch y EarlyStopping

13.1. train_epoch: Ciclo de entrenamiento

```
def train_epoch(model, loader, criterion, optimizer, device):
    model.train()  # Modo entrenamiento

    for X_batch, y_batch in loader:
        X_batch = X_batch.to(device)
        y_batch = y_batch.to(device)

        # 1. Forward pass
        outputs = model(X_batch)
        loss = criterion(outputs, y_batch)

        # 2. Backward pass
        optimizer.zero_grad()  # Limpiar gradientes
        loss.backward()        # Calcular gradientes

        # 3. Actualizar pesos
```

```
        optimizer.step()        # w = w - lr * grad

    return avg_loss, accuracy
```

13.2. Diagrama del ciclo



13.3. validate_epoch vs train_epoch

Diferencias:

- `model.eval()` en vez de `model.train()`
- `torch.no_grad()` para no calcular gradientes
- NO hay `optimizer.zero_grad()`, `loss.backward()`, `optimizer.step()`

13.4. EarlyStopping

Prevenir overfitting

Early Stopping detiene el entrenamiento cuando el modelo deja de mejorar en validacion. Guarda el mejor modelo y espera "patience" epocas antes de parar.

```
class EarlyStopping:
    def __init__(self, patience=15):
        self.patience = patience    # Epocas a esperar
        self.counter = 0            # Sin mejora
        self.best_loss = float('inf')
        self.best_state = None

    def __call__(self, val_loss, model):
        if val_loss < self.best_loss:
            # Mejora: guardar y resetear
            self.best_loss = val_loss
            self.best_state = model.state_dict().copy()
            self.counter = 0
        else:
            # No mejora: incrementar
            self.counter += 1
            if self.counter >= self.patience:
                return True  # Parar
        return False
```

14. Por que usamos esos hiperparametros y que es el F1-score

14.1. Nuestros hiperparametros

Hiperparametro	Valor	Justificacion
Learning rate	10^{-3}	Valor estandar para Adam
Weight decay	10^{-2}	Regularizacion L2 moderada
Batch size	64	Balance ruido/memoria
Dropout	0.3	30 % neuronas apagadas
Patience	15	Esperar mejoras lentas
Kernel size	5	Patrones de 0.1s

14.2. F1-Score explicado

Formula del F1-Score

$$F1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

Ejemplo: Clase SITTING

- TP = 408 (correctamente SITTING)
- FP = 79 (otras predichas como SITTING)
- FN = 92 (SITTING predicho como otra)

$$\text{Precision} = \frac{408}{408 + 79} = 0.837$$
$$\text{Recall} = \frac{408}{408 + 92} = 0.816$$
$$F1 = 2 \cdot \frac{0.837 \times 0.816}{0.837 + 0.816} = \boxed{0.827}$$

Por que media armonica?

La media armonica **penaliza valores extremos**. Si Precision = 1.0 pero Recall = 0.1:

- Media aritmetica: $(1.0 + 0.1)/2 = 0.55$ (parece “aceptable”)
- F1 (armonica): $2 \times 1.0 \times 0.1/1.1 = 0.18$ (revela el problema)

15. Como se calculan los parametros y que son los pesos

15.1. Que son los pesos y bias

Definiciones

- **Pesos (Weights)**: Valores que se multiplican por las entradas. Representan la importancia de cada conexion.
- **Bias**: Valor que se suma despues. Permite desplazar la funcion.
- **Parametros**: Todos los valores aprendibles (pesos + bias).

15.2. Formulas de conteo

Formulas

Conv1d:

$$\text{Params} = C_{out} \times (C_{in} \times K + 1)$$

BatchNorm1d:

$$\text{Params} = 2 \times C$$

Linear:

$$\text{Params} = \text{out} \times (\text{in} + 1)$$

15.3. Calculo detallado para nuestro modelo

Parametros del modelo HAR

Conv1d(9, 64, kernel=5): $64 \times (9 \times 5 + 1) = 64 \times 46 = \boxed{2,944}$
BatchNorm1d(64): $2 \times 64 = \boxed{128}$
Conv1d(64, 128, kernel=5): $128 \times (64 \times 5 + 1) = 128 \times 321 = \boxed{41,088}$
BatchNorm1d(128): $2 \times 128 = \boxed{256}$
Linear(4096, 256): $256 \times (4096 + 1) = 256 \times 4097 = \boxed{1,048,832}$
Linear(256, 6): $6 \times (256 + 1) = 6 \times 257 = \boxed{1,542}$
Total: $2,944 + 128 + 41,088 + 256 + 1,048,832 + 1,542 = \boxed{1,094,790}$

Nota: La capa Linear(4096, 256) tiene el 95.8 % de los parametros!

15.4. Como cambian los pesos

Regla de actualizacion

$$W_{nuevo} = W_{actual} - \eta \cdot \frac{\partial L}{\partial W}$$

- η = learning rate (10^{-3})
- $\frac{\partial L}{\partial W}$ = gradiente
- El signo negativo: vamos **opuesto** al gradiente (minimizamos)

Ejemplo numerico

Peso $W = 0.5$ con gradiente $\frac{\partial L}{\partial W} = 0.02$:

$$W_{nuevo} = 0.5 - (0.001)(0.02) = 0.5 - 0.00002 = 0.49998$$

El peso disminuyo ligeramente hacia el optimo.

16. Resumen: Los 15 Conceptos Clave

#	Concepto	Resumen
1	Semilla aleatoria	Garantiza reproducibilidad
2	Ejes X, Y, Z	3 dimensiones espaciales del telefono
3	50 Hz	50 mediciones por segundo
4	X e Y en ML	X = features, y = etiquetas
5	Sensores	body_acc, total_acc, body_gyro
6	Acc vs Gyro	Aceleracion lineal vs rotacion
7	Train/Test	Dividido por personas (no aleatorio)
8	Accuracy	Confiable si clases balanceadas
9	Z-score	$(x - \mu)/\sigma$; $\|r\| > 0.5$ = correlacion fuerte
10	Batch	Subconjunto de 64 muestras
11	Formulas	Conv1D, MaxPool, Linear, ReLU, Softmax
12	Forward Pass	Propagacion entrada $\rightarrow$ salida
13	Training	train_epoch actualiza; validate solo evalua
14	Hiperparametros	LR=10^{-3}, dropout=0.3, patience=15
15	Parametros	1,094,790 valores aprendibles

Mensaje Final

Este documento cubre los fundamentos necesarios para entender el proyecto HAR:

- Los **sensores** (2, 5, 6) generan los **datos** (3, 4)
- Los datos pasan por el **modelo** (11, 12, 15) durante el **entrenamiento** (10, 13)
- Usamos **hiperparametros** (14) y **reproducibilidad** (1) para obtener buenos resultados
- Evaluamos con **metricas** (8, 9) usando datos **separados correctamente** (7)