

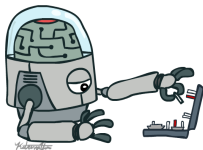
Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones



Aprendizaje Supervisado

A. Atutxa

LSI Bilbao

Noviembre 2023

¹Basado en el lecciones de Andrew Ng (Standford), Razvan Pascanu(DeepMind), Jay Alammal y Pieter Abbel (Berkeley)

Overview

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

1 Recapitulando sobre perceptrón

2 Mejorando la interpretación

3 Mejorando las actualizaciones

El Perceptron

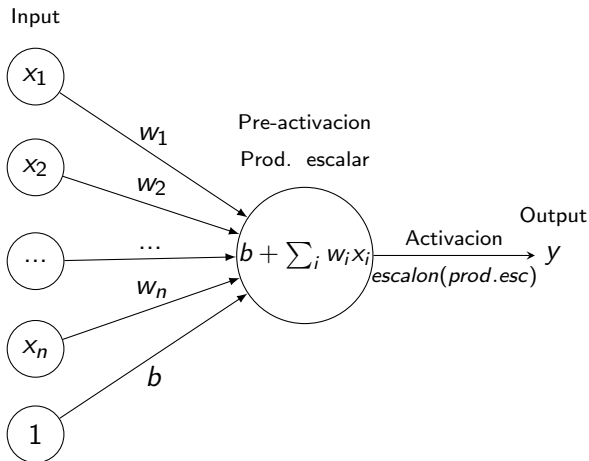
Intro AS

A. Atutxa

Rescapitulando
sobre
perceptr3n

Mejorando la
interpretaci3n

Mejorando las
actualiza-
ciones



Perceptrón: Algoritmo

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$w \in \mathbb{R}^n$, $b \in \mathbb{R}$, y^{ok} clase real $\in \{0, 1\}$, y^* clase predicha $\in \{0, 1\}$

Algorithm 1 pseudocódigo Perceptrón clasificación binaria

```
1:  $w = random()$ 
2: while  $\neg$ convergencia do
3:    $prodEsc = f(x) * w + b$ 
4:   clasificar  $\{1, 0\}$  dependiendo de  $prodEsc$ 
5:   actualizar  $w$  para mejorar predicción //Acerco—Alejo
6: end while
```

Perceptrón: Algoritmo

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$\mathbf{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$, y^{ok} clase real $\in \{0, 1\}$, y^* clase predicha $\in \{0, 1\}$

Algorithm 2 pseudocodigo en pasos genéricos Perceptrón clasificación binaria

```
1:  $w = random()$ 
2: while  $\neg$ convergencia do
3:    $prodEsc = f(x) * w + b$ 
4:   if  $prodEsc \geq 0$  then
5:      $y^* = 1$ 
6:   else
7:      $y^* = 0$ 
8:   end if
9:    $w = w + (y^{ok} - y^*) * f(x)$  //Acerco—Alejo según error
10: end while
```

Aprendizaje Supervisado: Estudio del precio del Chocolate

Intro AS

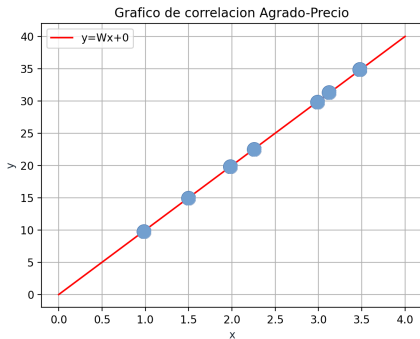
A. Atutxa

Rescapitulando sobre perceptrón

Mejorando la interpretación

Mejorando las actualizaciones

Nivel de Agrado	Dispuesto a Pagar
1	10
1.5	15
2	20
2.25	22.5
3	30
3.1	31
3.5	35
4	???



¿que valor está dispuesto a pagar el último cliente?
¿qué valor le corresponde a W ?

¹Ejemplo adaptado, fuente: Jay Alammar

Aprendizaje Supervisado: Estudio del precio del Chocolate

Intro AS

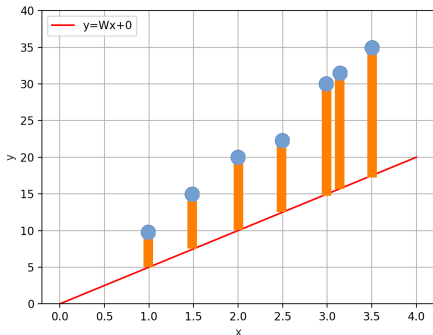
A. Atutxa

Rescapitulando sobre perceptrón

Mejorando la interpretación

Mejorando las actualizaciones

Nivel de Agrado	Dispuesto a Pagar
1	10
1.5	15
2	20
2.25	22.5
3	30
3.1	31
3.5	35
4	???



Si la inicialización random hubiese sido $W=5$ estaríamos cometiendo un error $(y - y^*)$. Y este error será el que guíe el aprendizaje como veremos

Perceptrón: Algoritmo

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$w \in \mathbb{R}^n$, $b \in \mathbb{R}$, y valor real $\in \mathbb{R}$, y^* valor predicho $\in \mathbb{R}$

Algorithm 3 pseudocodigo Regresión Lineal

```
1:  $w = \text{random}()$ 
2: while  $\neg$ convergencia do
3:    $\text{prodEsc} = f(x) * w + b$ 
      //en Regresion sobra la step function
4:   if  $\text{prodEsc} \geq 0$  then
5:      $y^* = 1$ 
6:   else
7:      $y^* = 0$ 
8:   end if
9:    $w = w + (y - y^*)^2 f(x)$  //Siempre acerco.
10: end while
```

Aprendizaje Supervisado: Escenario básico

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Siendo X el espacio de los valores de entrada.
- Siendo Y el espacio de los valores de salida.
- Para un determinado conjunto de datos $D \subset X \times Y$ encuentra la función h tal que
$$h : X \rightarrow Y$$
- h suele denominarse **hipótesis**
- La categorización del problema dependen del dominio de Y
 - si $Y \in \mathbb{R}$: es un problema de **regresión**
 - si $Y \in \text{val}1, \text{val}2 \dots \text{val}_n$: es un problema de **clasificación**

¹Transparencia de Razvan Pascanu

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- **Mejorando la función de activación** (en el caso de la clasificación)
- **Mejorando la actualización de pesos:** Evitar que sean bruscos, añadir generalización y obtener no solo un buen w sino el mejor.

Hacia la Regresión Logística: clasificación binaria

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$w \in \mathbb{R}^n$, $b \in \mathbb{R}$, y^{ok} clase real $\in \{0, 1\}$, y^* predicha $\in \{0, 1\}$

Algorithm 4 pseudocodigo a caballo entre Perceptrón y RLog

```
1:  $w = random()$ 
2: while  $\neg$ convergencia do
3:    $prodEsc = f(x) * w + b$ 
4:    $conf = \frac{1}{1 + e^{-prodEsc}}$  //sigmoide
5:   if  $conf \geq 0.5$  then
6:      $y^* = 1$ 
7:   else
8:      $y^* = 0$ 
9:   end if
10:   $w = w + (y - y^*) * f(x)$  //Aunque mejorable, luego  
    hablamos de cómo...
11: end while
```

Hacia la Regresión Logística: clasificación multiclase

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$w \in \mathbb{R}^n$, $b \in \mathbb{R}$, y^{ok} clase real $\in \mathbb{N}^k$, y^* clase predicha $\in \mathbb{N}^k$

Algorithm 5 pseudocod. a caballo entre Perceptrón y RLog multicl.

```
1:  $w = random()$ 
2: while  $\neg$ convergencia do
3:   for  $i = 1$  to  $k$  do
4:      $exp_i = e^{f(x) * w_i + b}$ 
5:      $softmax_i = \frac{exp_i}{\sum_1^k exp_i}$ 
6:   end for
7:    $y^* = \arg \max_i (softmax_i = \frac{exp_i}{\sum_1^k exp_i})$  // mantengo el resto de
   momento
8:   if  $y^* \neq y^{ok}$  then
9:      $w^{ok} = w^{ok} + f(x)$ 
10:     $w^* = w^* - f(x)$ 
11:   end if
12: end while
```

Funciones de Activación

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- sigmoide (σ_i) para clasificación binaria
- *softmax* para clasificación multiclass

Hacia la Regresión Logística: clasificación multiclase

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- ¿Puede ayudar actualizar los pesos aunque hayamos acertado?
- Es decir, ¿puede ayudar plantearlo como regresión intentando aprender la distribución de probabilidades de las clases?

Aprendizaje Supervisado: Regresión Logística

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Pred	Real	¿OK?
0.8	1	SI
0.9	1	SI
0.55	0	NO

Table: 2º Predictor

Pred	Real	¿OK?
0.6	1	SI
0.55	1	SI
0.6	0	NO

Table: 1º Predictor

No solo queremos medir si
hemos acertado o no

Queremos medir **cuánto** nos
hemos desviado y que **la**
desviación guíe nuestro
aprendizaje.

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Funciones medir la desviación (o error o coste): cross-entropy y error cuadrático (Mean Squared Error). Queremos minimizar nuestro error.

- **Cross-Entropy:**

- **Clasificación binaria:**

- $$-\frac{1}{N}(y_i \log(\sigma_i) + (1 - y_i) \log(1 - \sigma_i))$$

- **MSE:**

- **Clasificación binaria:** $\frac{1}{2N}(\sigma_i - y_i)^2$

Calculémoslo para el 1er ejemplo de los predictores anteriores 1 y 2

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

■ Primera red 1er item de entrenamiento:

- cross-entropy $:-\left(\frac{1}{N}(y_i \log(\sigma_i) + (1 - y_i) \log(1 - \sigma_i))\right)$
 $-(1 * \log(0.6)) + ((1 - 1) * \log(1 - 0.6)) =$
 $-\log(0.6) + 0 = 0.22/3 = 0.073$ (N = 3 instancias)

- MSE $\frac{1}{2N}(\sigma_i - y_i)^2$:
 $\frac{1}{2N}(0.6 - 1)^2 = \frac{1}{2N}(0.4^2) = 0.08/3 = 0.026$ (N = 3 instancias)

■ Segunda red 1er item de entrenamiento:

- cross-entropy $:-y_i \log(\sigma_i) + (1 - y_i) \log(1 - \sigma_i)$
 $-(1 * \log(0.8)) + ((1 - 1) * \log(1 - 0.8)) =$
 $-\log(0.8) + 0 = 0.09/3 = 0.03$

- MSE $\frac{1}{2N}(\sigma_i - y_i)^2$:
 $\frac{1}{2}(0.8 - 1)^2 = \frac{1}{2N}(0.2^2) = 0.02/3 = 0.006$

Aprendizaje Supervisado: Regresión Logística

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Pred			Real			¿OK?
0.1	0.2	0.7	0	0	1	SI
0.1	0.7	0.2	0	1	0	SI
0.3	0.4	0.3	1	0	0	NO

Table: 2º Predictor

Pred			Real			¿OK?
0.3	0.3	0.4	0	0	1	SI
0.3	0.4	0.3	0	1	0	SI
0.1	0.2	0.7	1	0	0	NO

Table: 1º Predictor

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Funciones medir la desviación (o error o coste): cross-entropy y error cuadrático (Mean Squared Error). Queremos minimizar nuestro error.

- **Cross-Entropy:**

- Clasificación **multiclase**: $\frac{-1}{N} \sum_{i=1}^{k=numClases} y_i \ln(\text{softmax}_i)$

- **MSE:**

- Clasificación **multiclase**: $\frac{1}{2N} \sum_{i=1}^{k=numClases} (\text{softmax}_i - y_i)^2$

Calculémoslo para el 1er ejemplo de los predictores anteriores 1 y 2

Cross-Entropy Binary vs Multiclass

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

■ Cross-Entropy (Binaria):

$$(y_i \log(\sigma_i) + (1 - y_i) \log(1 - \sigma_i))$$

■ Cross-Entropy (Multiclass, p.e. real [0,0,1]):

$$\begin{aligned} & \frac{-1}{N} \sum_{i=1}^{k=numClases} y_i \ln(\text{softmax}_i) \\ &= 0 * \ln\left(\frac{\exp_{z_1}}{\exp_{z_1} + \exp_{z_2} + \exp_{z_3}}\right) + 0 * \ln\left(\frac{\exp_{z_2}}{\exp_{z_1} + \exp_{z_2} + \exp_{z_3}}\right) + 1 * \\ & \ln\left(\frac{\exp_{z_3}}{\exp_{z_1} + \exp_{z_2} + \exp_{z_3}}\right) \end{aligned}$$

Pred (softmax_i)			Real			¿OK?
0.3	0.3	0.4	0	0	1	SI

Table: 1^{er} Predictor

Pred(softmax_i)			Real			¿OK?
0.1	0.2	0.7	0	0	1	SI

Table: 2^o Predictor

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

■ Primera red 1er item de entrenamiento:

- cross-entropy $\frac{1}{N} \sum_{i=1}^k -y_i \ln(\exp_i)$:
$$-((\ln(0.3) * 0) + (\ln(0.3) * 0) + (\ln(0.4) * 1)) = -\ln(0.4) = 0.9163/3$$
- MSE $\frac{1}{2M} \sum_{i=1}^k (\exp_i - y_i)^2$:
$$(0.3 - 0)^2 + (0.3 - 0)^2 + (0.4 - 1)^2 = 0.09 + 0.09 + 0.36 = \frac{1}{2} * 0.54/3$$

■ Segunda red 1er item de entrenamiento:

- cross-entropy:
$$-\frac{1}{N} * ((\ln(0.1) * 0) + (\ln(0.2) * 0) + (\ln(0.7) * 1)) = -\ln(0.7) = 0.3566/3$$
- MSE: $\frac{1}{2N} * ((0.1 - 0)^2 + (0.2 - 0)^2 + (0.7 - 1)^2) = \frac{1}{2N} * (0.01 + 0.04 + 0.09) = \frac{1}{2} * 0.14/3$

Tips

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- MSE: utilizada principalmente en tareas de regresión.
 - Errores por exceso o por defecto los trata igual al emplear el cuadrado.
- MSE: penaliza fuertemente los valores atípicos (outliers) dado que el error se vuelve cuadrático haciendo que los outliers sean fuertemente penalizados.
- Cross-Entropy para clasificación.

Más funciones de error

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Loss Function	Applicability to Classification	Applicability to Regression	Sensitivity to Outliers
Mean Squared Error (MSE)	✗	✓	High
Mean Absolute Error (MAE)	✗	✓	Low
Cross-Entropy	✓	✗	Medium
Hinge Loss	✓	✗	Low

Mejorando las predicciones y su interpretación

Intro AS

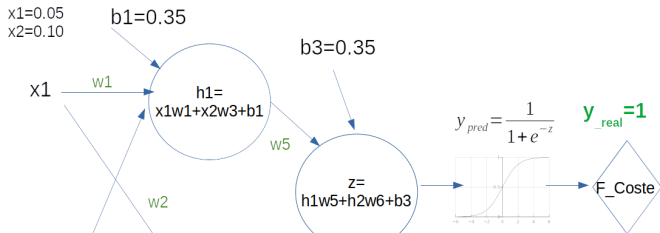
A. Atutxa

Recapitulando sobre perceptrón

Mejorando la interpretación

Mejorando las actualizaciones

Ejercicio clasificación binaria: predicción en alimentación hacia adelante (feed-forward)



F-Coste versión A:

$MSE =$

$$\frac{1}{2m} \sum_{i=1}^m (y_{real} - y_{pred})^2$$

m es el número de ejemplos.
Como solo tengo uno. $M=1$. Esto está unido al concepto Batch que trataremos después.

F-Coste versión B:

Binary_cross-entropy=

$$-\frac{1}{m} (y_{real} \log(y_{pred}) + (1 - y_{real}) \log(1 - y_{pred}))$$

Mejorando las predicciones y su interpretación

Intro AS

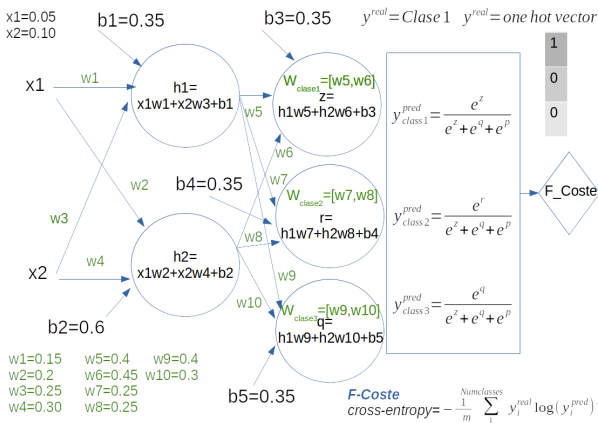
A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Ejercicio clasificación multiclase: predicción en alimentación hacia adelante (feed-forward)



Demo

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

<http://playground.tensorflow.org/>

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Actualización de pesos: Evitar que sean bruscos, añadir generalización y obtener no solo un buen w sino el mejor.

- **MIRA:** necesidad de añadir un elemento que minimice la actualización manteniendo el poder predictor. Concepto τ (similar al learning rate η)
- **Average Perceptrón:** Mejorar la generalización, suavizando el impacto de cada ejemplo sobre la actualización.
- ¿Se puede hacer aún mejor?

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Actualización de pesos: Evitar que sean bruscos, añadir generalización y obtener no solo un buen w sino el mejor.

- **MIRA:** necesidad de añadir un elemento que minimice la actualización manteniendo el poder predictor. Concepto τ (similar al learning rate η)
- **Average Perceptrón:** Mejorar la generalización, suavizando el impacto de cada ejemplo sobre la actualización.
- ¿Se puede hacer aún mejor?

Descenso del Gradiente (Gradient Descent)

Descenso del Gradiente

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Tengo una función que me mide la cantidad de error.
Cross-Entropy.
- Es una función derivable.
- La derivada mide la rapidez y la dirección del cambio de la función con respecto al cambio de una variable de esa función.

Descenso del Gradiente

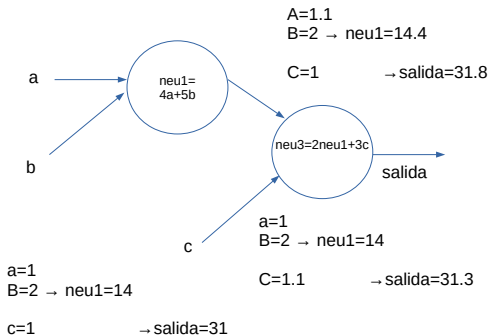
Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones



Derivada de la función coste

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$$f'(w) = \lim_{h \rightarrow 0} \frac{f(w+h) - f(w)}{h}$$

$$w \leftarrow w - \eta * \nabla_w \text{funcCoste}(w)$$

$$\begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix} - \eta * \begin{bmatrix} \frac{\partial \text{funcCoste}}{\partial w_1} \\ \frac{\partial \text{funcCoste}}{\partial w_2} \\ \dots \\ \frac{\partial \text{funcCoste}}{\partial w_n} \end{bmatrix}$$

Derivada de la función de coste

Intro AS

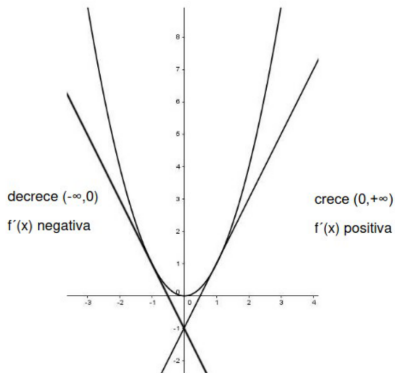
A. Atutxa

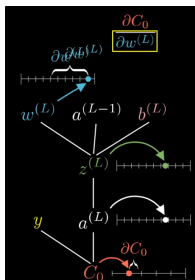
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$



Figure: Impacto del cambio¹

Un vídeo que explica muy bien la intuición es:
<https://www.youtube.com/watch?v=tIeHLnjs5U8>
 Visionarlo hasta el min. 3:20

¹Imagen de: 3Blue1Brown

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Queremos actualizar w_5 en base a lo que ha influido en la desviación que se ha producido. ¿Cómo medimos la "aportación de w_5 en la variación de la función de Coste"? La derivada mide eso, el ratio de cambio. $\frac{\delta \text{FuncCoste}}{\delta w_5}$

Se aplica la regla de la cadena:

$$\frac{\delta \text{FuncCoste}}{\delta w_5} = \frac{\delta \text{FuncCoste}}{\delta \sigma(z)} \frac{\delta \sigma(z)}{\delta z} \frac{\delta z}{\delta w_5}$$

Mejorando las predicciones y su interpretación

Intro AS

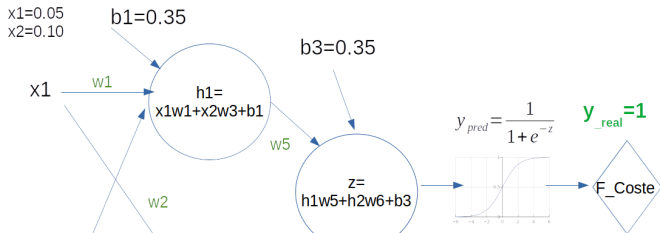
A. Atutxa

Recapitulando sobre perceptrón

Mejorando la interpretación

Mejorando las actualizaciones

Ejercicio clasificación binaria: predicción en alimentación hacia adelante (feed-forward)



F-Coste versión A:

$MSE =$

$$\frac{1}{2m} \sum_{i=1}^m (y_{real} - y_{pred})^2$$

m es el número de ejemplos.
Como solo tengo uno. $M=1$. Esto está unido al concepto Batch que trataremos después.

F-Coste versión B:

Binary_cross-entropy=

$$-\frac{1}{m} (y_{real} \log(y_{pred}) + (1 - y_{real}) \log(1 - y_{pred}))$$

Derivada de la función de coste

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$$\frac{d}{dx} c = 0$$

$$\frac{d}{dx} x^n = nx^{n-1}$$

$$\frac{d}{dx} e^{-x} = -e^{-x}$$

$$\frac{d}{dx} e^x = e^x$$

$$\frac{d}{dx} \ln x = \frac{1}{x}$$

$$\frac{d}{dx} \text{ numerador/denom} = (d(\text{num}) \cdot \text{den} - d(\text{den}) \cdot \text{num}) / \text{den}^2$$

Figure: Derivadas Básicas

Derivada de la función de Coste

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Si la función de coste hubiese sido MSE, la derivada sería $\frac{1}{2}2(y - \sigma(z))$
- Si la función de coste hubiese sido Binary Cross-Entropy, la derivada sería $-(\frac{y}{\sigma(z)} + \frac{1-y}{1-\sigma(z)})$

El resto de las derivadas son las mismas independientemente de la función de coste empleada. Es decir, $\frac{\delta \sigma(y)}{\delta y} \frac{\delta y}{\delta w_5}$

Derivada de la función sigmoide

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$$\frac{\delta\sigma(z)}{\delta z} = \frac{\frac{\delta \text{numerador}}{\delta(z)} * \text{denominador} - \frac{\delta \text{denominador}}{\delta(z)} * \text{numerador}}{(1+e^{-z})^2}$$

$$\frac{\delta\sigma(z)}{\delta z} = \frac{0*(1+e^{-z}) - (-e^{-z})*1}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2}$$

Truco para intentar simplificarlo en base a datos que ya hemos calculado en el forward

$$\begin{aligned} \frac{\delta\sigma(z)}{\delta z} &= \frac{e^{-z} + 1 - 1}{(1+e^{-z})^2} = \frac{e^{-z} + 1}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})} - \\ &\frac{1}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})} * \left(1 - \frac{1}{(1+e^{-z})}\right) = \sigma(z) * (1 - \sigma(z)) \end{aligned}$$

Derivada de la función sigmoide

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

$$\frac{\delta z}{\delta w_5} = h1$$
$$\left(-\frac{y}{\sigma(z)} + \frac{1-y}{1-\sigma(z)}\right) * (\sigma(z) * (1 - \sigma(z))) * h1$$

Mejorando las predicciones y su interpretación

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Como antes con la clasificación binaria, en la clasificación multiclase también queremos actualizar w_5 en base a lo que ha influido en la desviación que se ha producido. Igual que antes, la derivada mide eso, el ratio de cambio. $\frac{\delta F - Coste}{\delta w_5}$

Ejercicio: Backpropagation

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Vamos a ver cómo calcular la derivada de la función de coste que en este caso será la cross-entropy.

$$\text{Cross-entropy}^2: \sum_{i=1}^{k=\text{numClases}} -y_i \ln(\text{softmax}_i)$$

Para empezar la derivada de un sumatorio es el sumatorio de las derivadas. Hemos puesto un ejemplo concreto donde queremos derivar sobre w_5 para calcular la actualización de ese parámetro.

²Recordad que w_5 participa en el cálculo de e^{p_1} , e^{p_2} y e^{p_3} y estos en el cálculo de todos los softmax_i , en todos los denominadores ($e^{p_1} + e^{p_2} + e^{p_3}$) y en un numerador.

Derivada del Softmax

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Vamos a ver cómo calcular la derivada de la función de coste que en este caso será la cross-entropy.

$$\text{Cross-entropy}^3: \sum_{n=1}^{k=\text{numClases}} -y_i \ln(\text{softmax}_i)$$

Para empezar la derivada de un sumatorio es el sumatorio de las derivadas con su correspondiente regla de la cadena

$$\sum_{i=1}^{k=\text{numClases}} -y_i \frac{\delta \ln(\text{softmax}(p_i))}{\delta (\text{softmax}(p_i))} \frac{\delta \text{softmax}(p_i)}{\delta p_1} \frac{\delta (p_1)}{\delta w_5}$$

³Recordad que w_5 participa en el cálculo de e^{p_1} , e^{p_2} y e^{p_3} y estos en el cálculo de todos los softmax_i , en todos los denominadores ($e^{p_1} + e^{p_2} + e^{p_3}$) y en un numerador.

Derivada del Softmax

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Vamos a ver cómo calcular la derivada de la función de coste que en este caso será la cross-entropy.

$$\text{Cross-entropy: } \sum_{n=1}^{k=\text{numClases}} -y_i \ln(\text{softmax}_i)$$

Para empezar la derivada de un sumatorio es el sumatorio de las derivadas con su correspondiente regla de la cadena

$$\sum_{i=1}^{k=\text{numClases}} -y_i \frac{\delta \ln(\text{softmax}(p_i))}{\delta (\text{softmax}(p_i))} \frac{\delta \text{softmax}(p_i)}{\delta p_1} \frac{\delta (p_1)}{\delta w_5}$$

Derivada del Softmax

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Sabiendo que la derivada de $\ln(x) = \frac{1}{x}$

$$\sum_{i=1}^{k=numClases} \frac{-y_i}{softmax(p_i)} \frac{\delta softmax(p_i)}{\delta p_1} \frac{\delta(p_1)}{\delta w_5}$$

queda saber 1. $\frac{\delta softmax(p_i)}{\delta p_1}$ y 2. $\frac{\delta(p_1)}{\delta w_5}$

■ Respecto al 1. \rightarrow la derivada de $\frac{\delta softmax(p_i)}{\delta p_1}$

Cuando $i = 1$ será $\frac{\delta(\frac{e^{p_1}}{e^{p_1} + e^{p_2} + e^{p_3}})}{\delta p_1}$

Cuando $i \neq 1$ será $\frac{\delta(\frac{e^{p_{j=2,3}}}{e^{p_1} + e^{p_2} + e^{p_3}})}{\delta p_1}$

Derivada del Softmax

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Cuando $i = 1$ la derivada de $\frac{\delta \text{softmax}(p_i)}{\delta p_1}$ será $\frac{\delta(\frac{e^{p_1}}{(e^{p_1} + e^{p_2} + e^{p_3})})}{\delta p_1}$

Aplicando la regla del cociente

$$\begin{aligned} \frac{\delta(\frac{e^{p_1}}{(e^{p_1} + e^{p_2} + e^{p_3})})}{\delta p_1} &= \frac{\frac{\delta \text{numerador}}{\delta(p_1)} * \text{denominador} - \frac{\delta \text{denominador}}{\delta(p_1)} * \text{numerador}}{(\text{denominador})^2} \\ &= \frac{e^{p_1} * (e^{p_1} + e^{p_2} + e^{p_3}) - e^{p_1} * e^{p_1}}{(e^{p_1} + e^{p_2} + e^{p_3})^2} = \\ &= \frac{e^{p_1}}{(e^{p_1} + e^{p_2} + e^{p_3})} * \frac{(e^{p_1} + e^{p_2} + e^{p_3}) - e^{p_1}}{(e^{p_1} + e^{p_2} + e^{p_3})} \\ &= \text{softmax}(p_1) * (1 - \text{softmax}(p_1)) \end{aligned}$$

Cuando $i \neq 1$ la derivada de $\frac{\delta(\frac{e^{p_j=2,3}}{(e^{p_1} + e^{p_2} + e^{p_3})})}{\delta p_1}$

Derivada del Softmax

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Cuando $i \neq 1$ la derivada de $\frac{\delta softmax(j=2,3)}{\delta p_1}$ será

$$\frac{\delta\left(\frac{e^{p_{j=2,3}}}{(e^{p_1}+e^{p_2}+e^{p_3})}\right)}{\delta p_1}$$

Aplicando la regla del cociente i fijando $j=2$ (con $j=3$ será igual)

$$\begin{aligned}\frac{\delta\left(\frac{e^{p_{j=2,3}}}{(e^{p_1}+e^{p_2}+e^{p_3})}\right)}{\delta p_1} &= \frac{\frac{\delta numerator}{\delta(p_1)} * denominador - \frac{\delta denominator}{\delta(p_1)} * numerator}{(denominator)^2} \\ &= \frac{0 * (e^{p_1} + e^{p_2} + e^{p_3}) - e^{p_1} * e^{p_2}}{(e^{p_1} + e^{p_2} + e^{p_3})^2} = - \frac{e^{p_1}}{(e^{p_1} + e^{p_2} + e^{p_3})} * \frac{e^{p_2}}{(e^{p_1} + e^{p_2} + e^{p_3})} \\ &= -softmax(p_1) * softmax(p_2)\end{aligned}$$

Derivada del Softmax

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

Juntándolo todo:

$$\left(\sum_{i=2}^{k=numClases} \frac{-y_i}{softmax(p_i)} * -softmax(p_1) * softmax(p_i) \right) + \frac{-y_1}{softmax(p_1)} * softmax(p_1)(1 - softmax(p_1)) \frac{\delta(p_1)}{\delta w_5}$$

- Los softmax redundantes se eliminan

$$\left(\sum_{i=2}^{k=numClases} -y_i * softmax(p_1) \right) - y_1(1 - softmax(p_1)) \frac{\delta(p_1)}{\delta w_5}$$

- el $softmax(p_1)$ se saca fuera del sumatorio

$$(softmax(p_1) \sum_{i=2}^{k=numClases} -y_i) - y_1(1 - softmax(p_1)) \frac{\delta(p_1)}{\delta w_5}$$

- y recordar que $\sum_{i=1}^{k=numClases} -y_i = 1$

- y de esto se sigue que $\sum_{i=2}^{k=numClases} -y_i = 1 - y_1$

- Reescribiendolo quedaría:

$$(softmax(p_1) * (1 - y_1) - y_1(1 - softmax(p_1))) \frac{\delta(p_1)}{\delta w_5}$$

- Simplificando: $(softmax(p_1) - y_1) \frac{\delta(p_1)}{\delta w_5}$

Actualización por lotes (batch), Estocástica o por Minilotes

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

¿Cuándo realizo la actualización?

- por cada item de entrenamiento (Stochastic Gradient Descent). Desventaja: No me permiten vectorizar
- al final de un epoch (Batch Gradient Descent) Desventaja: Si tengo muchos datos hasta el final de cada epoch no actualizo.
- cada J ejemplos (Mini-batch Gradient Descent) Ventaja: Es un compromiso entre los dos anteriores

Se recomienda visualizar el vídeo de Andrew Ng colgado en el servidor. <http://lsi.bp.ehu.es/asignaturas/TIA/>

Más cuestiones técnicas

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Dropout
- ReLu
- Learning Rate
- Diferencia entre parámetros e hiperparámetros: Conjunto de validación versus conjunto de test

Aspectos técnicos importantesi

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

■ Dropout

El sobreajuste es un problema grave a medida que vamos añadiendo capas a nuestra red y neuronas en cada capa. El dropout es una técnica que permite abordar este problema. La idea clave es **desactivar aleatoriamente** neuronas (junto con sus conexiones) durante el entrenamiento. Así, conseguimos que todas las neuronas no se entrenen con todos los ejemplos de entrenamiento dado que se habrán desactivado con alguno de los ejemplos. Es como simulásemos distintos entrenamientos para cada neurona dado que cuando se ha desactivado no se ha entrenado sobre ese ejemplo.

Aspectos técnicos importantes

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

■ ReLU: Introduciendo no linealidad.

Las unidades ReLu introducen no linealidad, permitiendo la adaptación a cualquier contorno no lineal. Se define como $\max(0, x)$. Mirar explicación del enunciado del labo

Más cuestiones técnicas

Intro AS

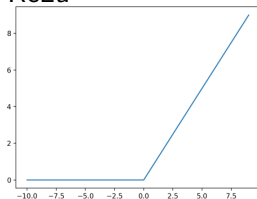
A. Atutxa

Recapitulando
sobre
perceptrón

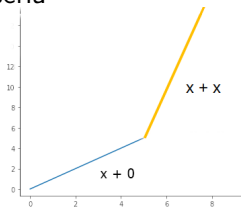
Mejorando la
interpretación

Mejorando las
actualiza-
ciones

ReLu



Supongamos que conectamos la salida x a una unidad ReLu de forma $x + \text{ReLu}(x-5)$, entonces la nueva salida sería



Aspectos técnicos importantes

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Learning Rate o Tasa de Aprendizaje α (otras veces se le llama η)

La tasa de aprendizaje es un hiperparámetro que permite **regular** cuánto ajustamos los pesos de nuestro modelo.

$$w_i = w_i + \alpha \left(\frac{\delta F - Coste}{\delta w_5} \right)$$

Recordad que queremos evitar actualizaciones bruscas. Hay que hacer un barrido de valores normalmente entre 0.1 y 0.0001 para encontrar el valor que nos permita regular mejor la actualización

Aspectos técnicos importantes

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Diferencia entre parámetros e hiperparámetros:
 - Los parámetros son los pesos que vamos actualizando AUTOMATICAMENTE a través de las iteraciones.
 - Los hiperparámetros son los valores que nos permiten generar distintos modelos. Por ejemplo, el número de iteraciones, el porcentaje de dropout, el learning-rate....

Aspectos técnicos importantes

Intro AS

A. Atutxa

Resumen
Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- El número de iteraciones es un hiperparámetro, a veces en vez de decir explícitamente cuantas iteraciones (epochs) se harán sobre el train, se marca cuantas iteraciones sin cambio en la tasa de error para decidir cuando parar.
- El learning rate es otro hiperparámetro cuyos valores a probar suelen ir de 0.1 a 0.0001

Haremos un barrido de estos hiper parámetros para generar distintos modelos. Si vemos que el número de combinaciones es muy grande, se puede hacer un barrido random, es decir generar combinaciones de forma random y una vez que tengamos la mejor combinación, afinar el barrido.

Aspectos técnicos importantes

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Cada modelo generado con los barridos debe ser evaluado. Para eso es necesario disponer de **set de validación** (para medir que modelo es el mejor). Es importante que también exista un **set de test** (el set que me permitirá hacer la evaluación definitiva y honesta) que permitirá a otros científicos o modelos compararse con el que se ha construido.

Aspectos técnicos importantes

Intro AS

A. Atutxa

Recapitulando
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

- Conclusión si queremos ajustar hiperparámetros:
Dividiremos siempre que tengamos hiperparámetros
nuestros datos en **3 conjuntos** train, validation y test.
Hay que asegurarse de que la división se hace de manera
random y que se mantiene la distribución de los datos
originales.

Métodos como *StratifiedShuffleSplit* de sklearn nos
permiten obtener los sets. *StratifiedShuffleSplit* es una
combinación de *ShuffleSplit* y *StratifiedKFold* de forma
que la proporción de distribución de etiquetas de clase es
casi uniforme entre los conjuntos generados.

Vídeos interesantes

Intro AS

A. Atutxa

Resumiendo
sobre
perceptrón

Mejorando la
interpretación

Mejorando las
actualiza-
ciones

En la url <http://lsi.bp.ehu.es/asignaturas/TIA/> encontraréis vídeos sobre el gradiente y otros conceptos que se han trabajado en este tema. Los vídeos son de Andrew Ng que es uno de los precursores de Coursera y de los cursos de ML. Os recomiendo que los véais.
<https://www.youtube.com/watch?v=tIeHLnjs5U8>