Procesamiento de Lenguaje Natural

Clase 10 – Intro Redes Neuronales (I)

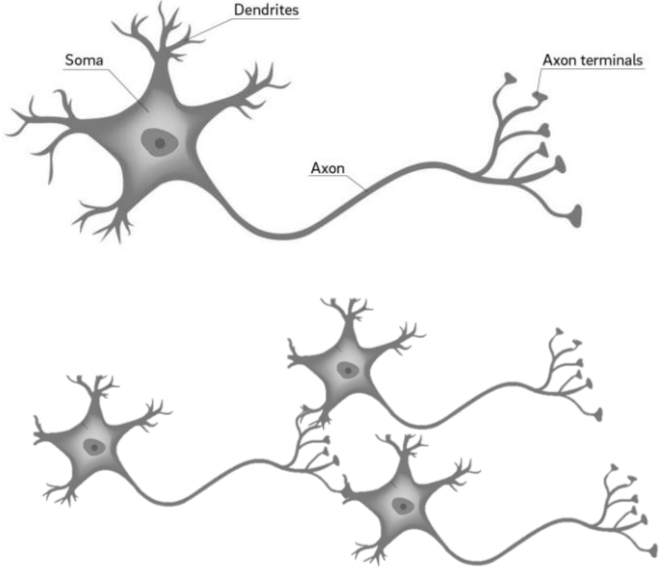
Ph.D. Rubén Manrique

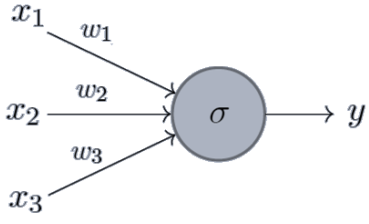
rf.manrique@uniandes.edu.co

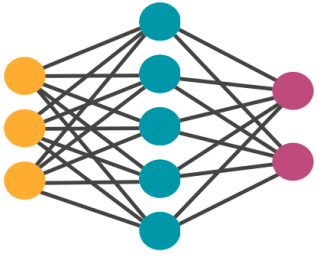
Maestría en Ingeniería de Sistemas y Computación

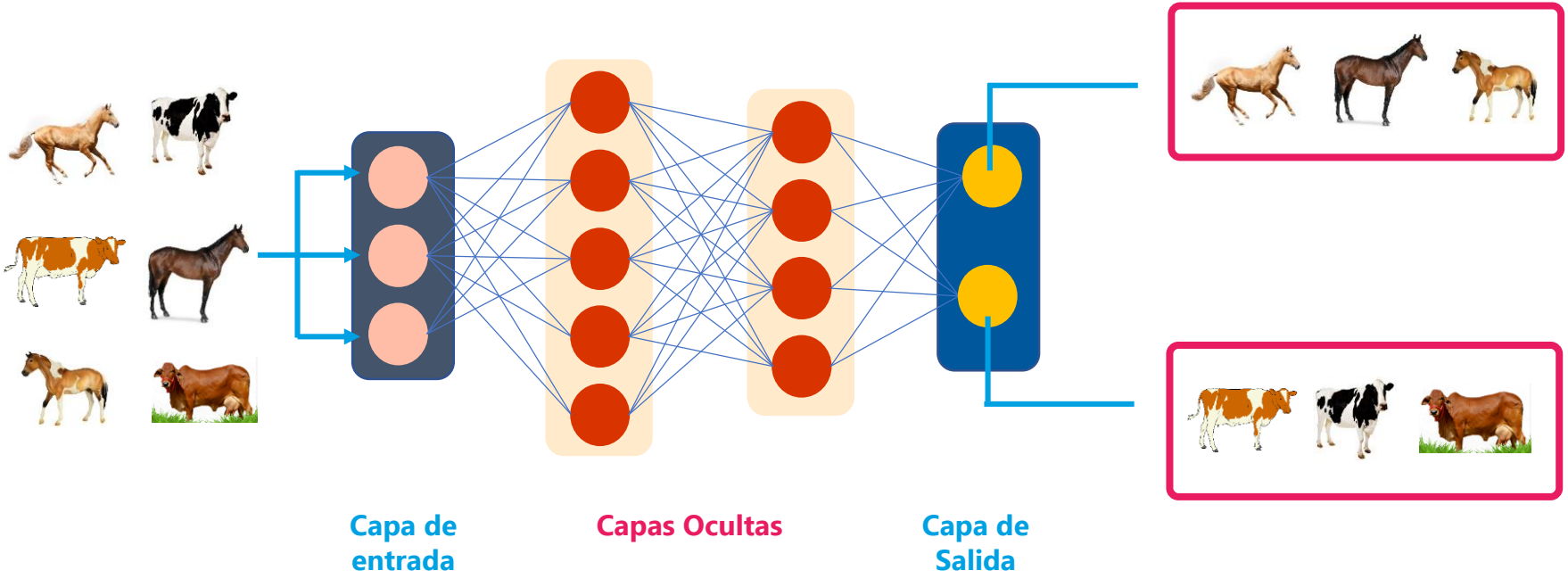
- Chapter 7: Speech and Language Processing

Redes Neuronales Artificiales: Inspiración Biológica.








Capa de entrada
Capas Ocultas
Capa de Salida

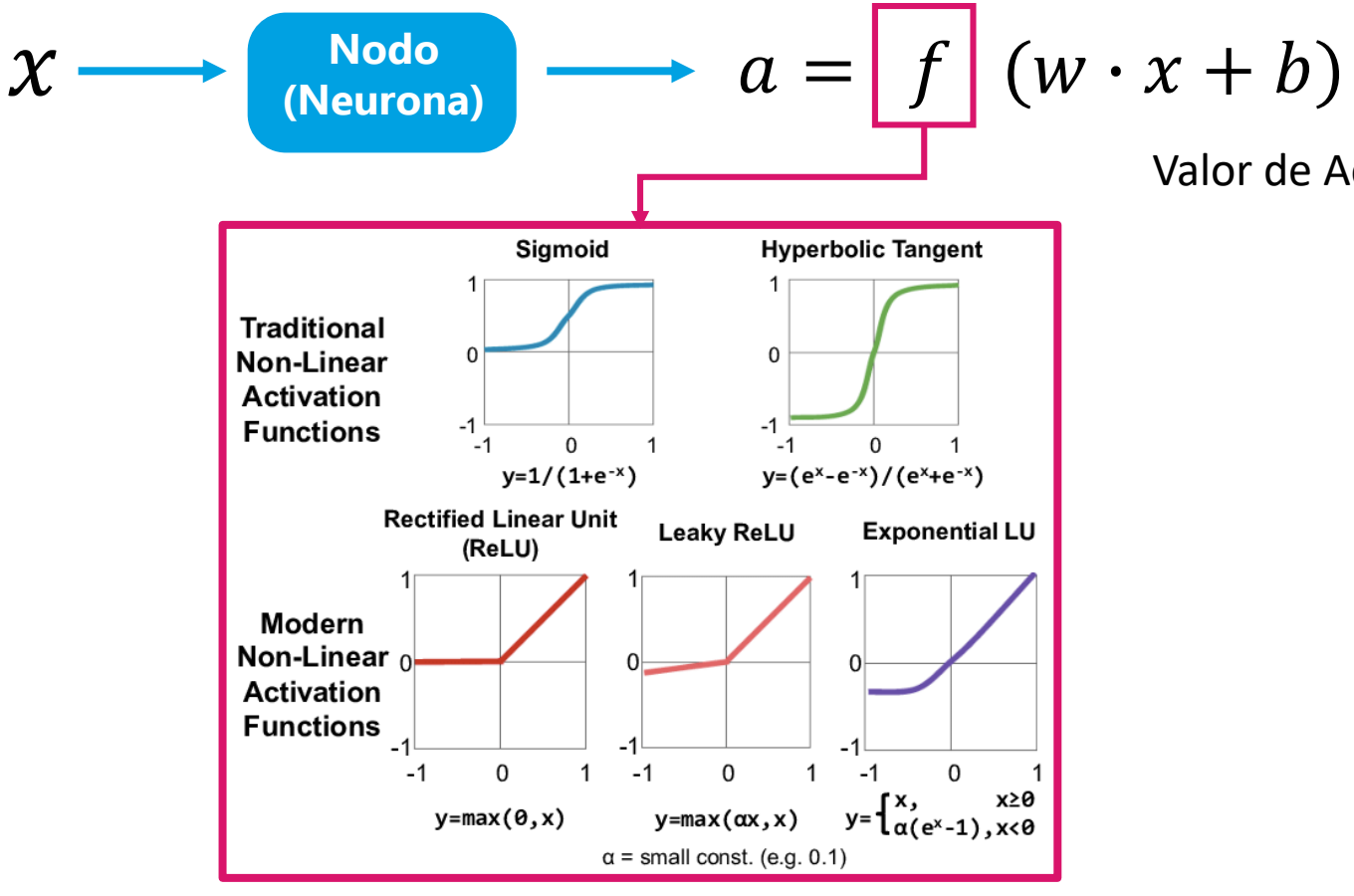
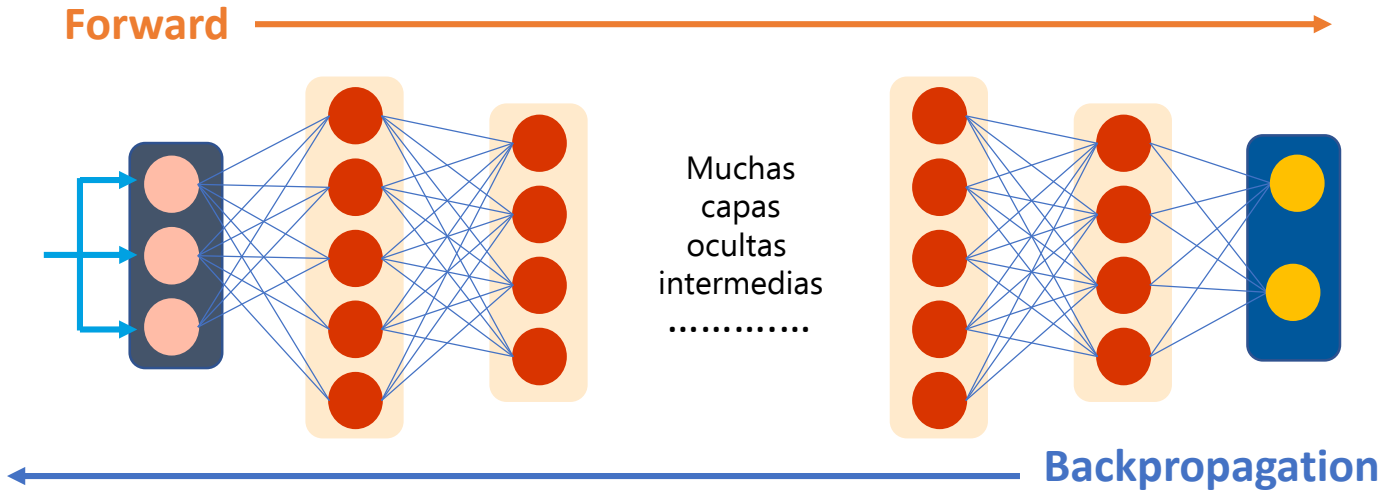


Imagen tomado de Caffe Tutorial

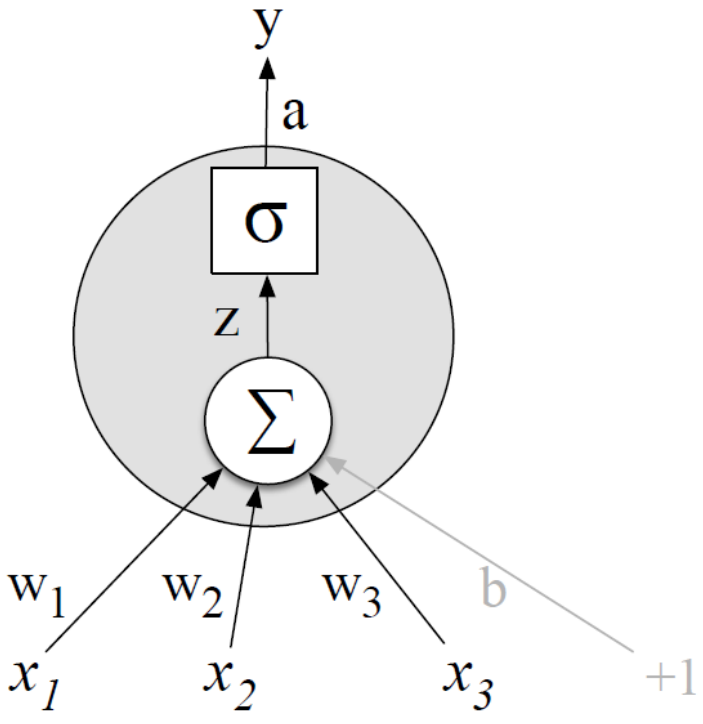
Redes Profundas



Cada uno de los pesos de la red neuronal recibe una actualización proporcional a la derivada parcial de la función de error con respecto al peso actual en cada iteración del entrenamiento.

PROBLEMA DE GRADIENTE DESVANECIENTE

Unit (Perceptron)



$$y = \sigma(w \cdot x + b) = \frac{1}{1 + \exp(-(w \cdot x + b))}$$

w = [0.2,0.3,0.9]

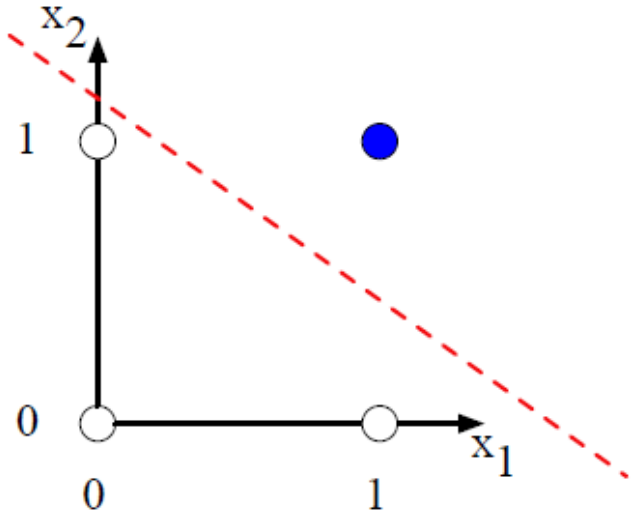
x = [0.5,0.6,0.1]

b = 0.5

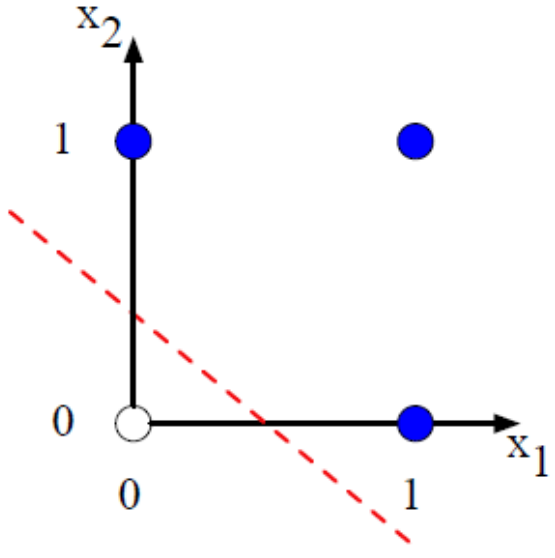
$$y = \frac{1}{1 + e^{-(.5*.2+.6*.3+.1*.9+.5)}}$$

$$y = \frac{1}{1 + e^{-(0.87)}} = 0.7$$

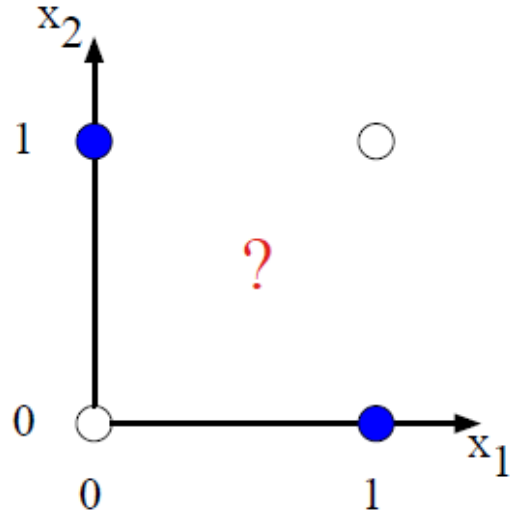
Perceptrón: Solo problemas linealmente separables



a) x_1 AND x_2

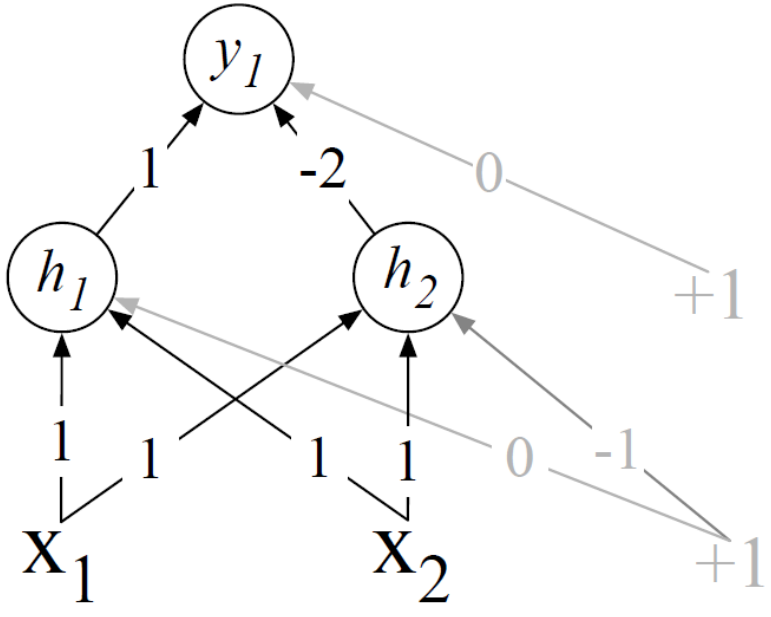


b) x_1 OR x_2



c) x_1 XOR x_2

Sol: Capas de Unidades – Redes Neuronales

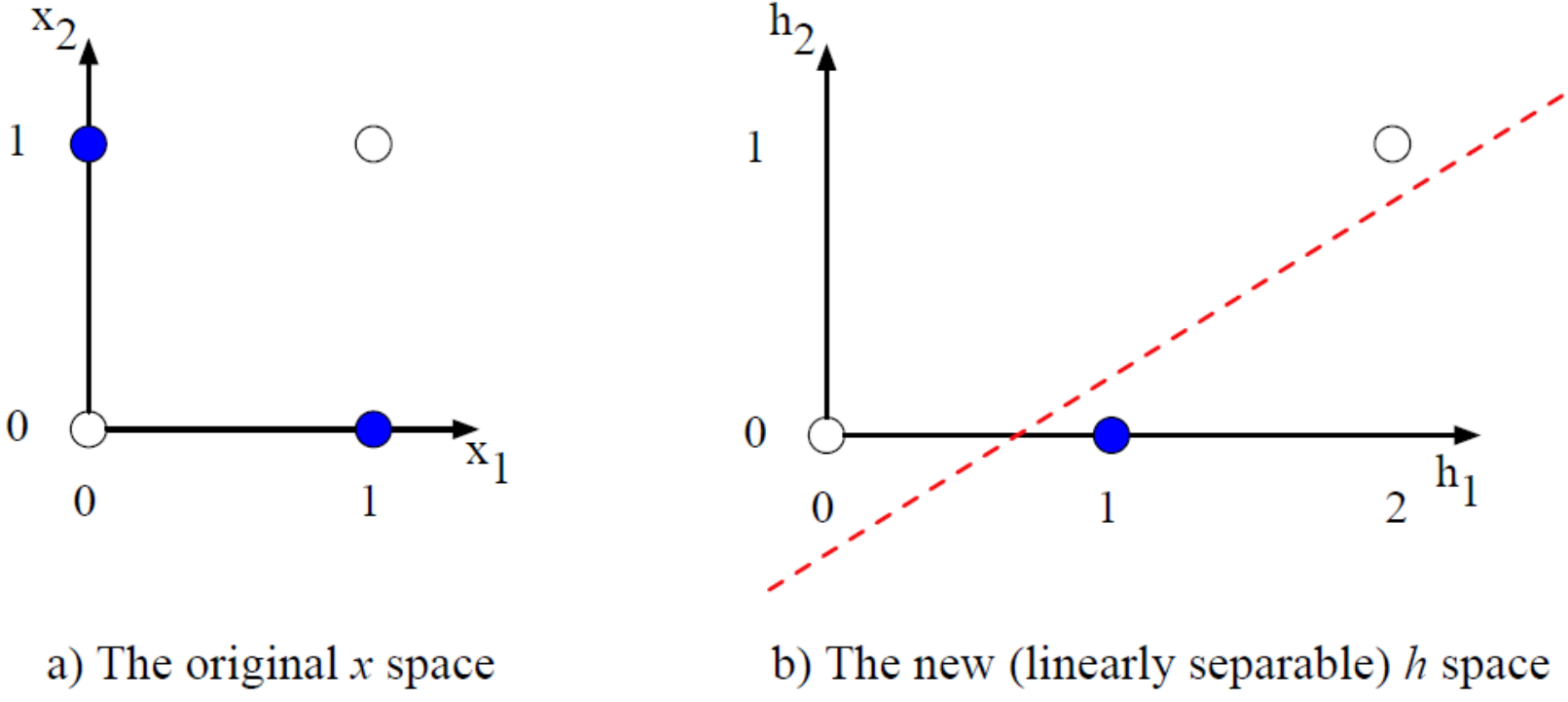


3 Unidades: ReLU.
2 Unidades en la capa oculta (h1+h2)
1 Unidad de salida (y)

x = [0,0]
y1 = ?
h1 = max(0,0)
h2 = max(0,-1)
h = [0,0]
y1 = max(0,0) = 0

x = [1,0]
h1 = max(1,0)=1,
h2 = max(0,-1)=0,
y1= max(1,0)=1

Capas Ocultas: Nueva representación de las entradas.

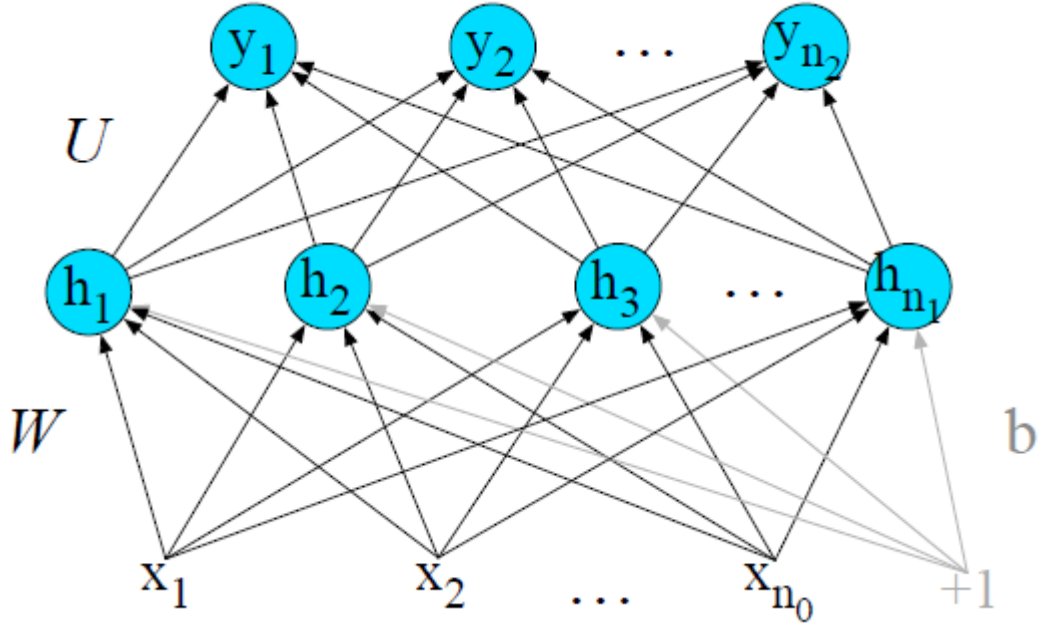


a) The original *x* space

b) The new (linearly separable) *h* space

Nuevas representaciones de las entradas ≈ Genera las *features*

Feed-Forward Neural Networks, also MLP



- Las salidas de cada unidad/nodo son pasados a los nodos de la siguiente capa.
- Normalmente totalmente conectada.
- Cada unidad/nodo conectada sin ciclos.
- Cada unidad/nodo tiene sus parámetros w y b.

Weight matrix W

- Representamos los parámetros de una capa oculta como la combinación del vector de pesos (w_i) y el bias (b_i) de cada unidad i mediante una matriz de pesos $\boldsymbol{W}$ y un vector de bias $\boldsymbol{b}$.
- Cada elemento $\boldsymbol{W_{ij}}$ representa el peso de la conexión de la ith entrada $\boldsymbol{x_i}$ a la unidad/nodo $\boldsymbol{h_j}$.
- La salida de la capa oculta el vector $\boldsymbol{h}$ es entonces calculado como:

vector

$$\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

La función de activación es aplicada a cada elemento del vector.

Dimensiones y notación

- El vector de entradas **x** es de n_0 dimensiones (0 denota la capa de entrada): $\boldsymbol{x} \in \mathbb{R}^{n_0}$ (vector columna $[n_0,1]$).
- La capa oculta tiene dimensionalidad n_1 por lo tanto $\mathbf{h} \in \mathbb{R}^{n_1}$ y $\mathbf{b} \in \mathbb{R}^{n_1}$.
- La matriz de pesos **W** por lo tanto: $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n_0}$ $([n_1, n_0])$.

$$\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

$$\begin{pmatrix} w_{1,1} & \cdots & w_{1,n_0} \\ \vdots & \ddots & \vdots \\ w_{n_1,1} & \cdots & w_{n_1,n_0} \end{pmatrix} \begin{bmatrix} x_1 \\ \cdots \\ x_{n_0} \end{bmatrix} + \begin{bmatrix} b_1 \\ \cdots \\ b_{n_1} \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + \cdots + w_{1,n_0}x_{n_0} + b_1 \\ \cdots \\ w_{n_1,1}x_1 + \cdots + w_{n_1}x_{n_0} + b_{n_1} \end{bmatrix}$$

Y la capa de salida?

- Si estamos haciendo una tarea binaria como la clasificación de sentimientos, podríamos tener una sola nodo de salida, y su valor y es la probabilidad de sentimiento positivo versus negativo.

- Si estamos haciendo una clasificación multinomial, como asignar una etiqueta POS, podría tener un nodo de salida para cada parte del discurso potencial.

Capa de salida representación

- La capa de salida puede o no estar con el parámetro bias, pero si tiene pesos. Denotaremos los pesos de la capa de salida como la matriz $\boldsymbol{U} \in \mathbb{R}^{n_2 \times n_1}$, la salida $\boldsymbol{z} \in \mathbb{R}^{n_2}$ se calcula como:

$$\boldsymbol{z} = \boldsymbol{U}\boldsymbol{h}$$

- Sin embargo, $\boldsymbol{z}$ no puede ser la salida del clasificador, ya que es un vector de valores reales números, mientras que lo que necesitamos para la clasificación es un vector de probabilidades.

- **Softmax:** codifica una distribución de probabilidad.

$$softmax(z_i) = \frac{e^{zi}}{\sum_{j=1}^{K} e^{z_j}}$$

Para el ejemplo $K = n_2$?

2-layer NN ecuaciones

$$\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

$$\boldsymbol{z} = \boldsymbol{U}\boldsymbol{h}$$

$$\mathbf{y} = \boldsymbol{softmax}(\boldsymbol{z})$$

Y redes mas profundas? - Notación

- Usaremos superíndices entre corchetes para indicar los números de capa, comenzando en 0 para la capa de entrada.
  - Entonces, $\boldsymbol{W}^{[1]}$y $\boldsymbol{b}^{[1]}$ significará la matriz de pesos y el vector de bias para la primera capa oculta.
  - n_j será el número de unidades/nodos de la capa j.
  - Vamos a usar $g()$ para referirnos a la función de activación.
    - ReLU or tanh para las capas intermedias (ocultas).
    - Softmax para las capas de salida.
  - $\boldsymbol{a}^{[i]}$ significa la salida de la capa i.
  - $\boldsymbol{z}^{[i]}$ es la combinación lineal de pesos, entradas y bias: $\boldsymbol{z}^{[i]} = \boldsymbol{W}^{[i]}\boldsymbol{a}^{[i-1]} + \boldsymbol{b}^{[i]}$
  - $\boldsymbol{a}^{[i]} = g^{[1]}(\boldsymbol{z}^{[i]})$
  - El vector de entrada $\boldsymbol{x} = \boldsymbol{a}^{[0]}$

2-layer NN ecuaciones nueva notación

- $\boldsymbol{z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{a}^{[0]} + \boldsymbol{b}^{[1]}$
- $\boldsymbol{a}^{[1]} = g^{[1]}(\boldsymbol{z}^{[1]})$
- $\boldsymbol{z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$
- $\boldsymbol{a}^{[2]} = g^{[2]}(\boldsymbol{z}^{[2]})$
- $\hat{y} = \boldsymbol{a}^{[2]}$

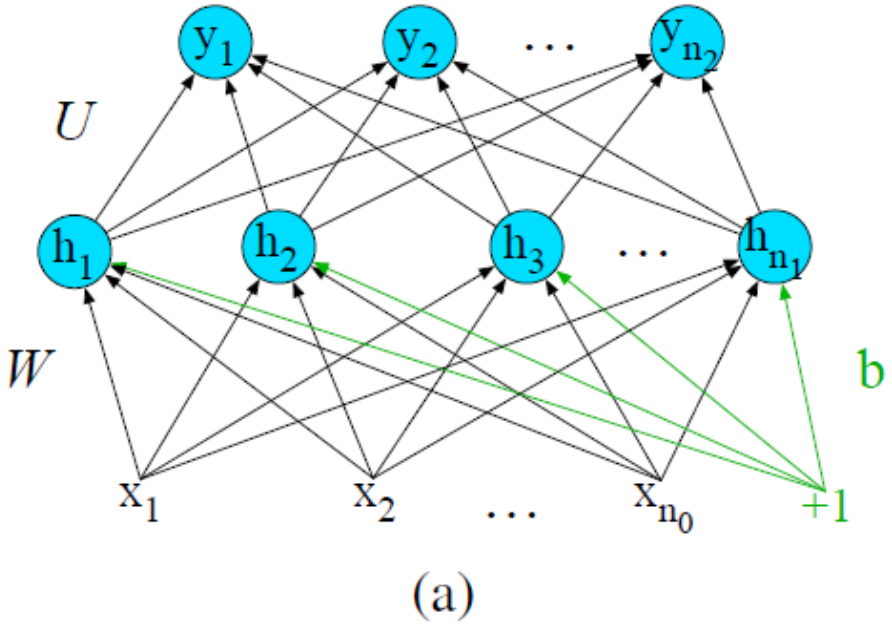
Forward Step (Generic Computation)

Given $\boldsymbol{a}[\boldsymbol{0}]$
$For\ \boldsymbol{i}\ in\ \boldsymbol{1}..\boldsymbol{n}$:
$\boldsymbol{z}^{[i]} = \boldsymbol{W}^{[i]}\boldsymbol{a}^{[i-1]} + \boldsymbol{b}^{[i]}$
$\boldsymbol{a}^{[i]} = g^{[i]}(\boldsymbol{z}^{[i]})$
$\hat{y} = \boldsymbol{a}^{[n]}$

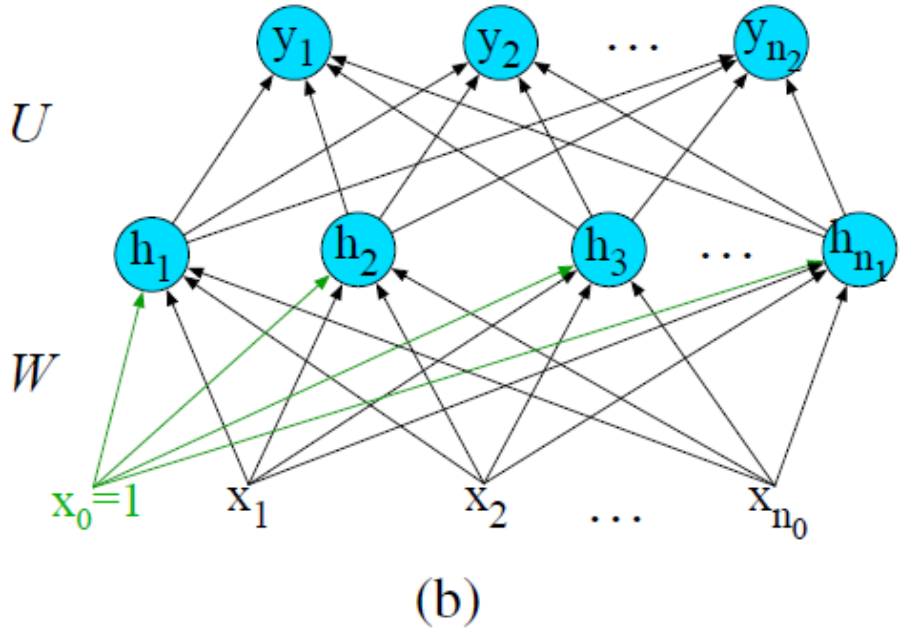
Simplificación del bias (I)

- Por simplicidad es mejor incorporar $\boldsymbol{b}$ como un valor adicional en la matrix **W.**
  - Se adiciona un nodo ficticio a_0 a cada capa y su valor se mantiene en 1. $a_0^{[0]} = 1, a_0^{[1]} = 1 \ldots$. Este nodo ficticio tendrá un peso que representará el valor del bias b.
  - Por lo tanto:
    - $\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ se reemplaza por $\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x})$

Simplificación del bias (II)



(a)



(b)

Overfitting

Training data

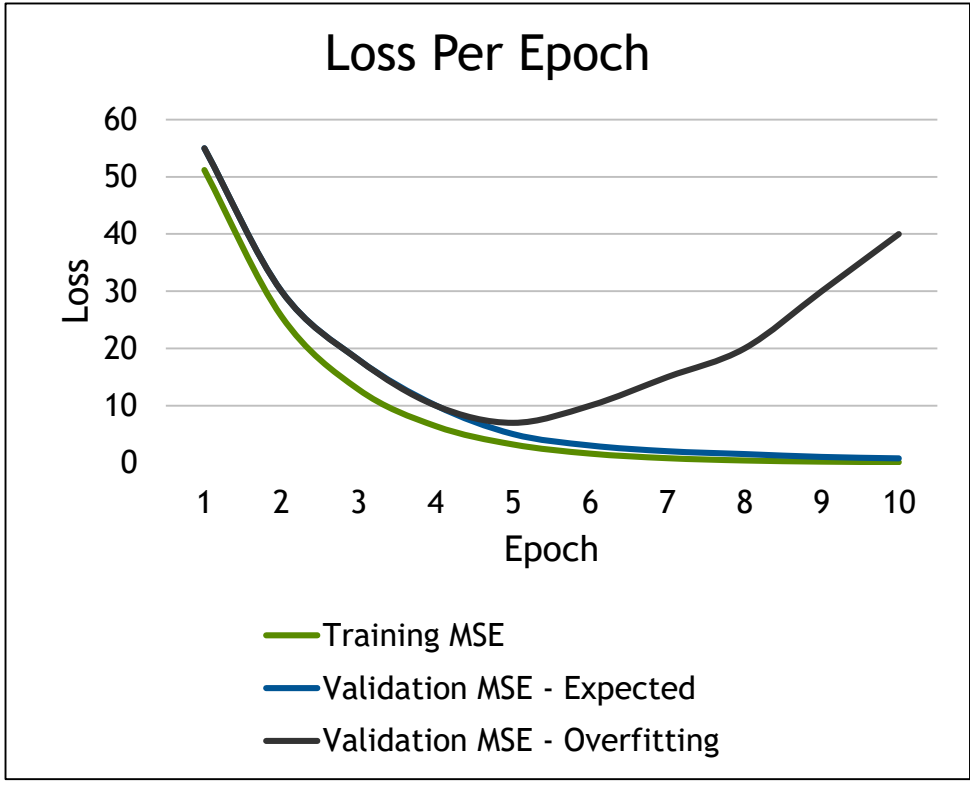
- Core dataset for the model to learn on

Validation data

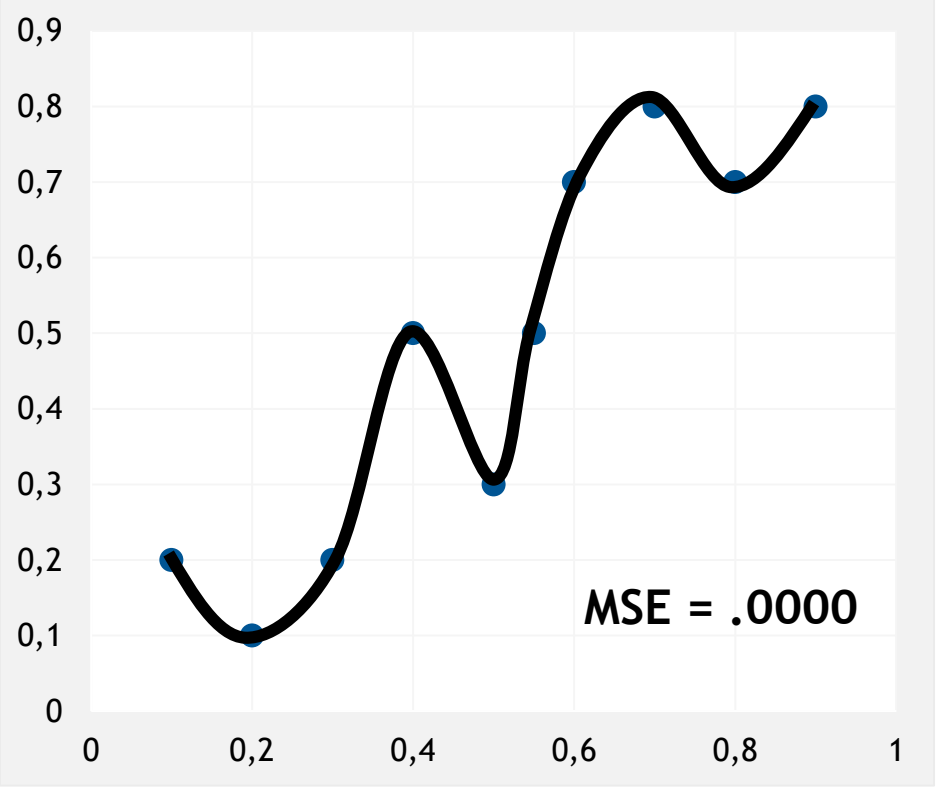
- New data for model to see if it truly understands (can generalize)

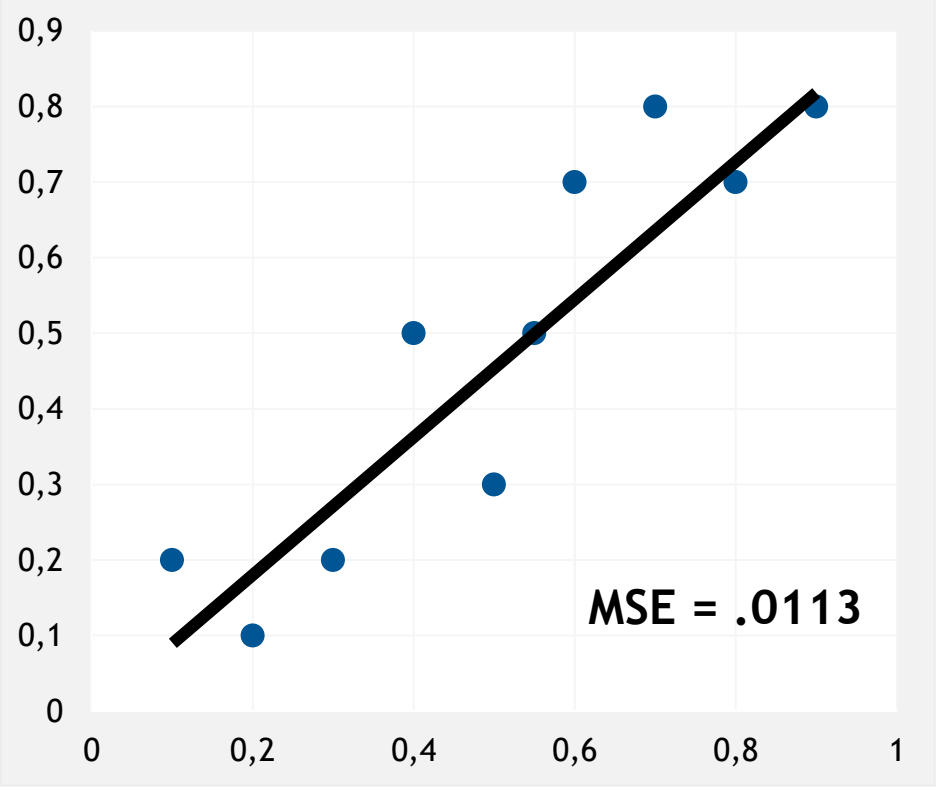
Overfitting

- When model performs well on the training data, but not the validation data (evidence of memorization)
- Ideally the accuracy and loss should be similar between both datasets

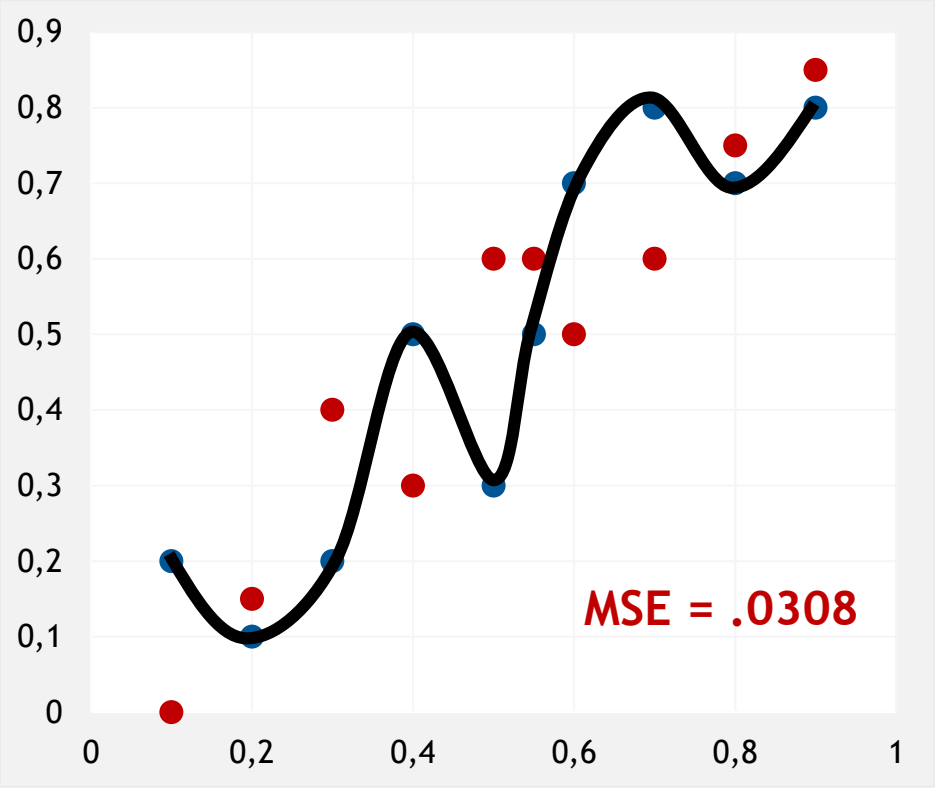


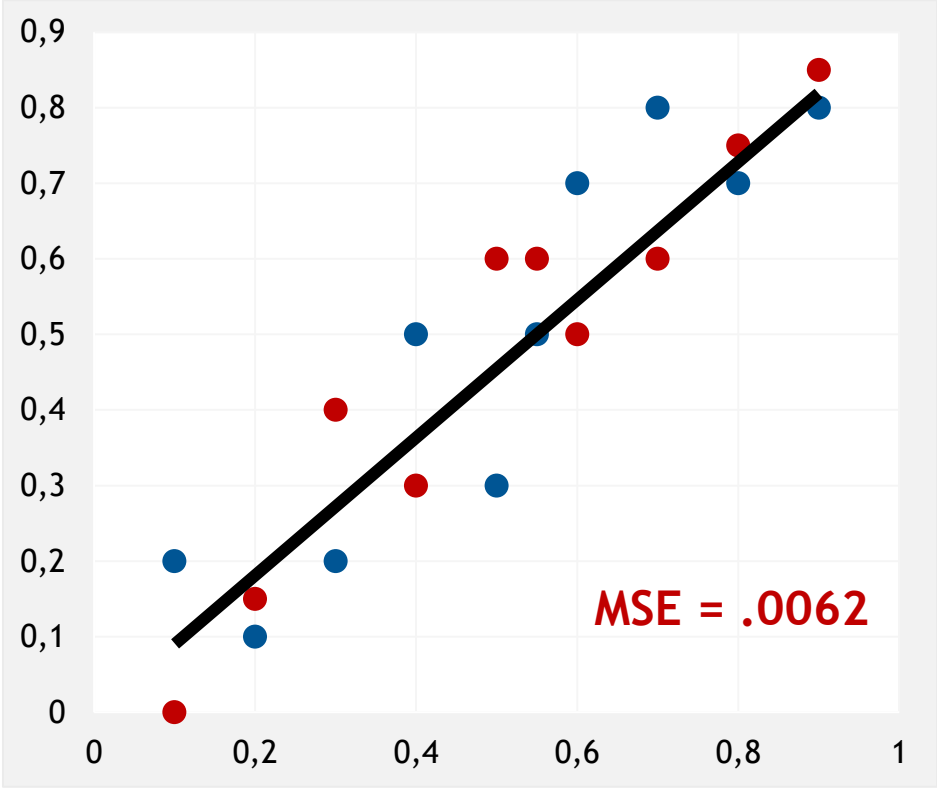
Overfitting (Training)



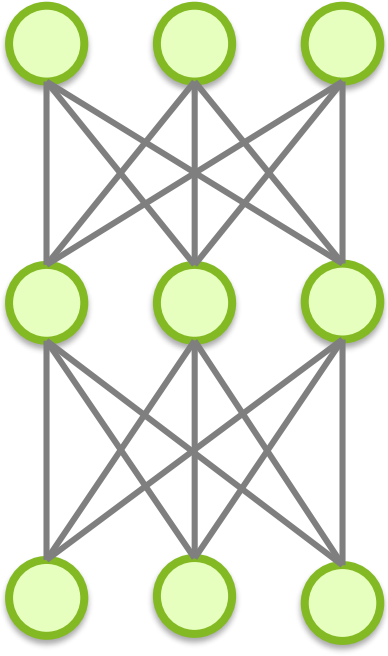


Overfitting (Testing)

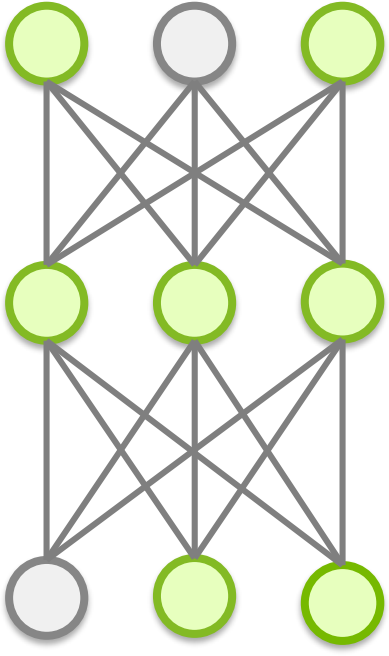




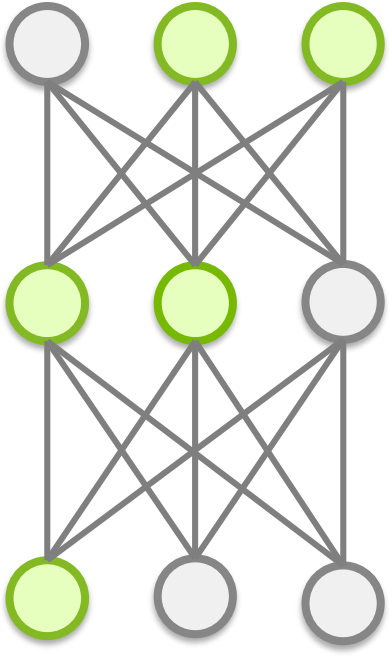
Dropout



rate = 0



rate = .2



rate = .4

Entrenamiento de redes neuronales

- El objetivo del proceso de entrenamiento es estimar los parámetros $\boldsymbol{W}^{[i]}$ para cada capa i que hace la estimación ($\hat{y}$) de cada ejemplo de entrenamiento lo mas cercana posible a la verdad y.
  - **Loss function:** cross-entropy loss.
  - **Método de optimización**: gradient descent.
  - Requerimos el **gradiente de la loss function**, es decir, el vector que contiene las derivadas parciales con respecto a cada uno de los parámetros.
  - La principal diferencia con la regresión logística radica en este punto. Quien me la puede indicar?
  - Para responder a esta necesidad se creo el algoritmo de **error backpropagation**.

Cross-entropy loss

$$L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1-y) \log(1-\hat{y})]$$

Para el multinomial (C clases):

$$L_{CE}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log \hat{y_i}$$

Si solo una clase k de las C es correcta para cada ejemplo (hard classification task) y y se representa como one-hot vector:

$$L_{CE}(\hat{y}, y) = -\mathbf{1}\{\mathrm{y} = \mathrm{k}\} \log \hat{y_i}$$

$$L_{CE}(\hat{y}, y) = -\log \hat{y_i} \quad para\ i = k$$

$$L_{CE}(\hat{y}, y) = -\log \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

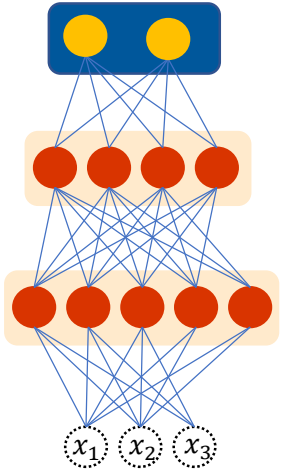
Calculando el Gradiente (recordando de LR)

- $\frac{\partial L_{CE}(w,b)}{\partial w_j} = (\hat{y} - y)x_j = (\sigma(wx + b) - y)x_j$
- $\frac{\partial L_{CE}(w,b)}{\partial w_j} = \left(\mathbf{1}\{\mathrm{y} = \mathrm{k}\} - \frac{e^{(w_k x + b_k)}}{\sum_{j=1}^{K} e^{(w_j x + b_j)}}\right) x_j$

Capa de Salida

Capas Ocultas

Entradas



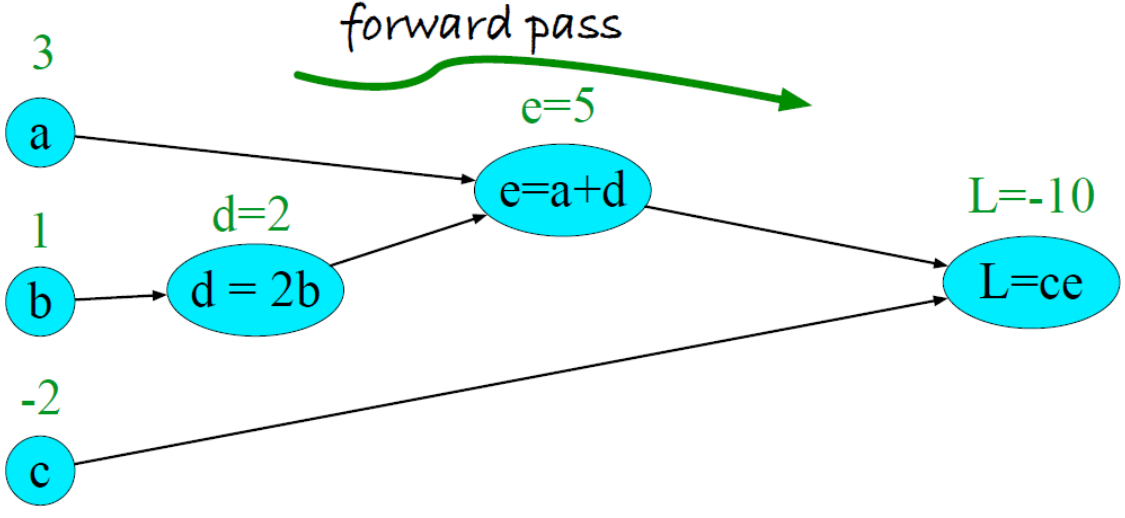
Las anteriores derivadas funcionan para actualizar los pesos de que capa?

Solución: Backprop (1986, Rumelhart).

Backward differentation
Computation Graphs

Computation Graphs

- Un grafo de computo es una representación del proceso de cálculo de una expresión, en la que el cálculo se divide en operaciones separadas, cada de los cuales se modela como un nodo en un gráfico.
  - Considere la función L(a;b;c) = c(a+2b).



Backward differentiation on computation graphs (I)

- El objetivo es calcular la derivada de la función L con respecto a cada una de esas variables $\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial c}$.
- Regla de la cadena en calculo. Suponga una función compuesta $f(x) = u\big(v(x)\big)$, la derivada de $f(x)$ por regla de la cadena es:

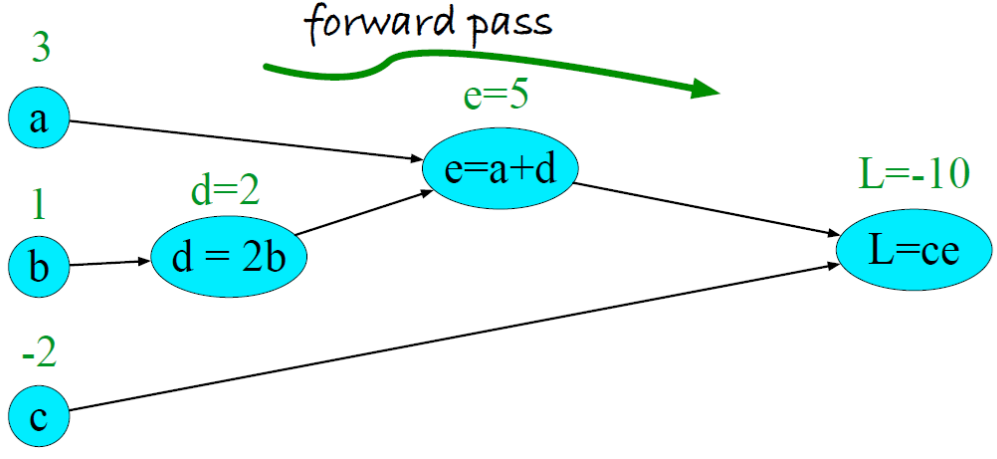
$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx}$$

- Extendida para mas de dos funciones:

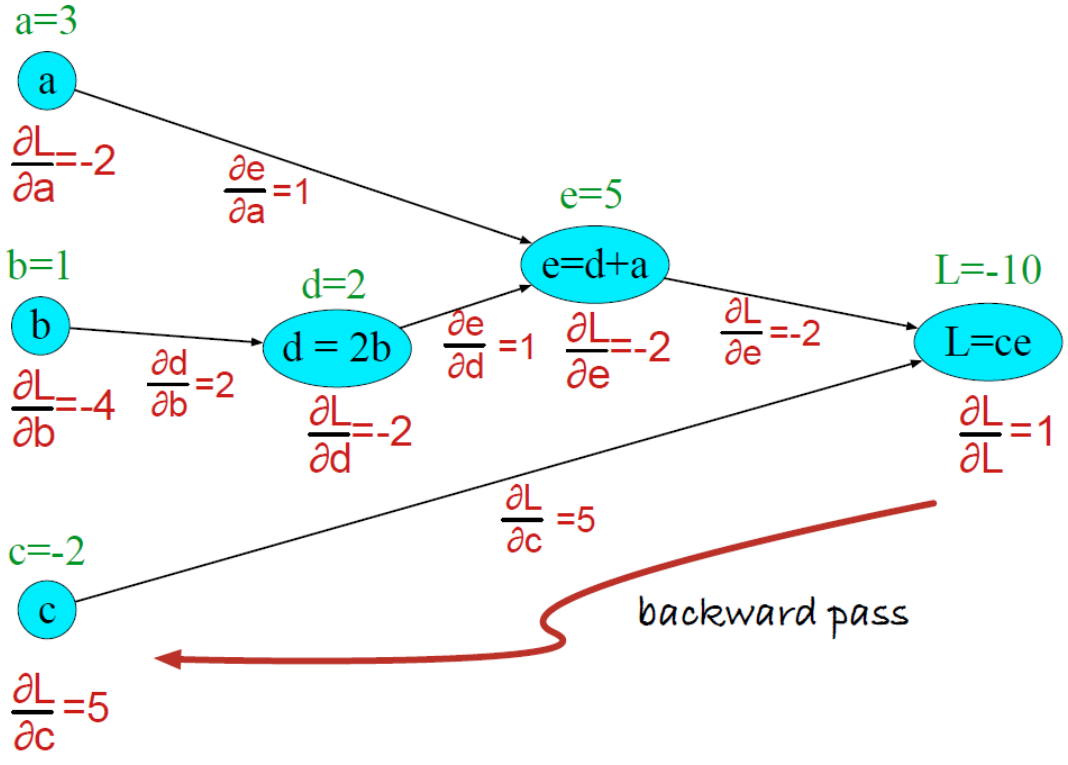
$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dw} \cdot \frac{dw}{dx}$$

Backward differentiation on computation graphs (II)

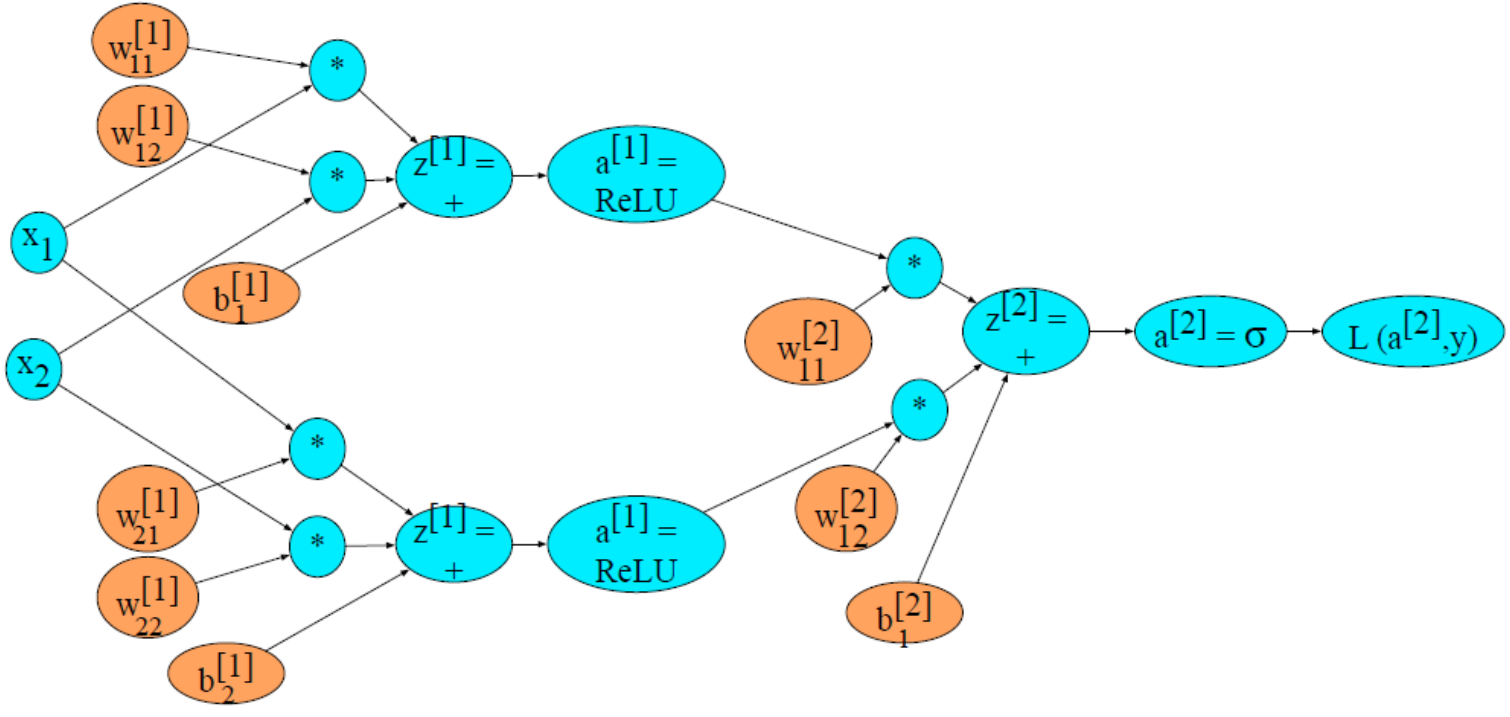
- $\frac{\partial L}{\partial c} = \mathrm{e} = 5$
- $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial a} = (\mathrm{c}) * (1) = -2$
- $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial b} = (c) * (1) * (2) = 4$



Backward differentiation on computation graphs (III)



- $\boldsymbol{z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}}$
- $\boldsymbol{a^{[1]} = g^{[1]}(z^{[1]})}$
- $\boldsymbol{z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}}$
- $\boldsymbol{a^{[2]} = g^{[2]}(z^{[2]})}$
- $\hat{y} = a^{[2]}$



Que derivadas necesitamos?

- $\frac{d\sigma(z)}{dz} = \sigma(z)\big(1 - \sigma(z)\big)$
- $\frac{dtanh(z)}{dz} = (1 - tanh^2(z))$
- $\frac{dReLU(z)}{dz} = \begin{cases} 0 \;\; for \;\; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$
- Después del forward aplicamos la diferenciación hacia atrás en el grafo de computo.

Diferencias con RL

- La optimización en redes neuronales es un problema de optimización no convexa.

- Para la regresión logística, podemos inicializar el descenso de gradiente con todos los pesos y sesgos que tengan el valor 0. En las redes neuronales, por el contrario, necesitamos inicializar los pesos con números aleatorios pequeños (**pregunta de examen**).

- También es útil normalizar los valores de entrada para que tengan una media de 0 y una varianza unitaria (convergencia lenta por posibles rangos no comparables).

Detalles importantes en entrenamiento

- Para evitar overfitting: usar regularización por dropout. Aleatoriamente se eliminan unidades/nodos y sus conexiones de la red durante el entrenamiento.

- El ajuste de los hiperparámetros también es importante: learning rate η, mini-batch size, la arquitectura del modelo. La misma selección del optimizador (hay muchas varientes de implementación del gradiente descendiente, por ejemplo Adam).

Modelos de Lenguaje Neuronales

Neural Language Models

- Ventajas sobre los modelos de lenguaje basados en n-gramas:
  - No requieren smoothing.
  - Pueden manejar contextos mucho mayores.
  - Pueden generalizar mejor la “similaridad” de palabras sobre diversos contextos.

- Modelos:
  - Feedforward neural language.
  - Recurrent neural language.

Feedforward neural language (I)

- Es una red feedforward estándar
  - Toma como entrada a un tiempo t una representación de algunos de las palabras previas ($w_{t-1}, w_{t-2}, etc.$).
  - A partir de estas entradas la red estima la probabilidad de una palabras a partir del contexto $P(w_t|w_1:w_{t-1})$ utilizando la N palabras previas:

$$P(w_t|w_1:w_{t-1}) \approx P(w_t|w_{t-N+1}, \dots, w_{t-1})$$

  - Si $N = 4$

$$P(w_t|w_1:w_{t-1}) \approx P(w_t|w_{t-1}, w_{t-2}, w_{t-3})$$

Feedforward neural language (II)

- **Nota importante**: “El contexto es representado por **embeddings** de las palabras previas”.

- Representar el contexto anterior como un embedding, en lugar de exactamente palabras utilizadas en los modelos de lenguaje n-gram, permite capacidad de generalización a datos no visto previos previamente.

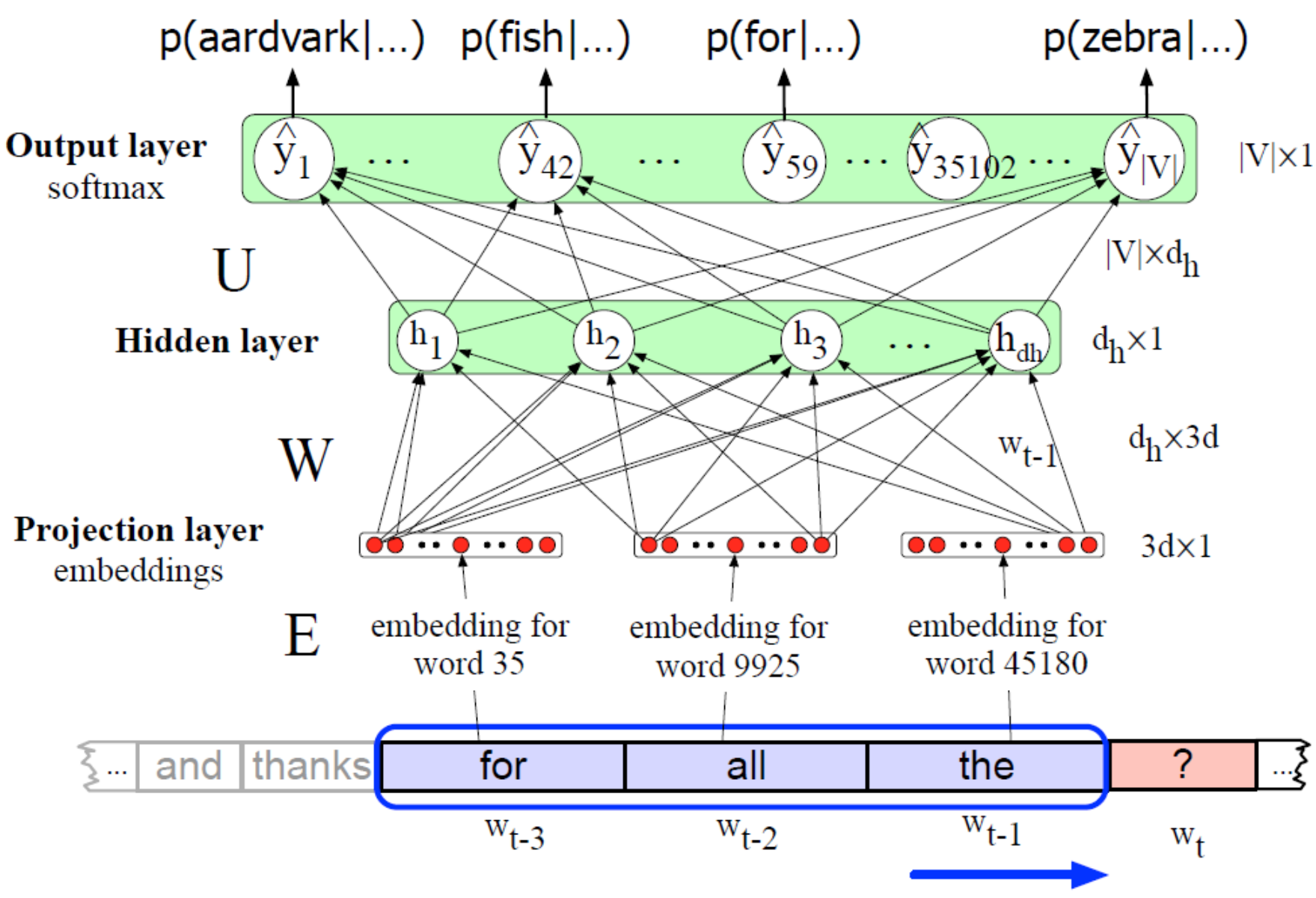
Capacidad de generalización con embeddings (I)

- Suponga que la siguiente sentencia fue vista en el entrenamiento:
  - I have to make sure that the cat gets fed.
- Suponga además que en el dataset de entrenamiento nunca se evidenciaron las palabras “gets fed” después de la palabra “dog”.

- En el dataset de prueba “I forgot to make sure that the dog gets ___”.
  - Que pasaría con un modelo de lenguaje basado en n-gramas sin smoothing.
  - Con embeddings “cat” and “dog” debería tener representaciones similares, y es posible que asigne fed aun cuando “dog” nunca fue visto en el training .

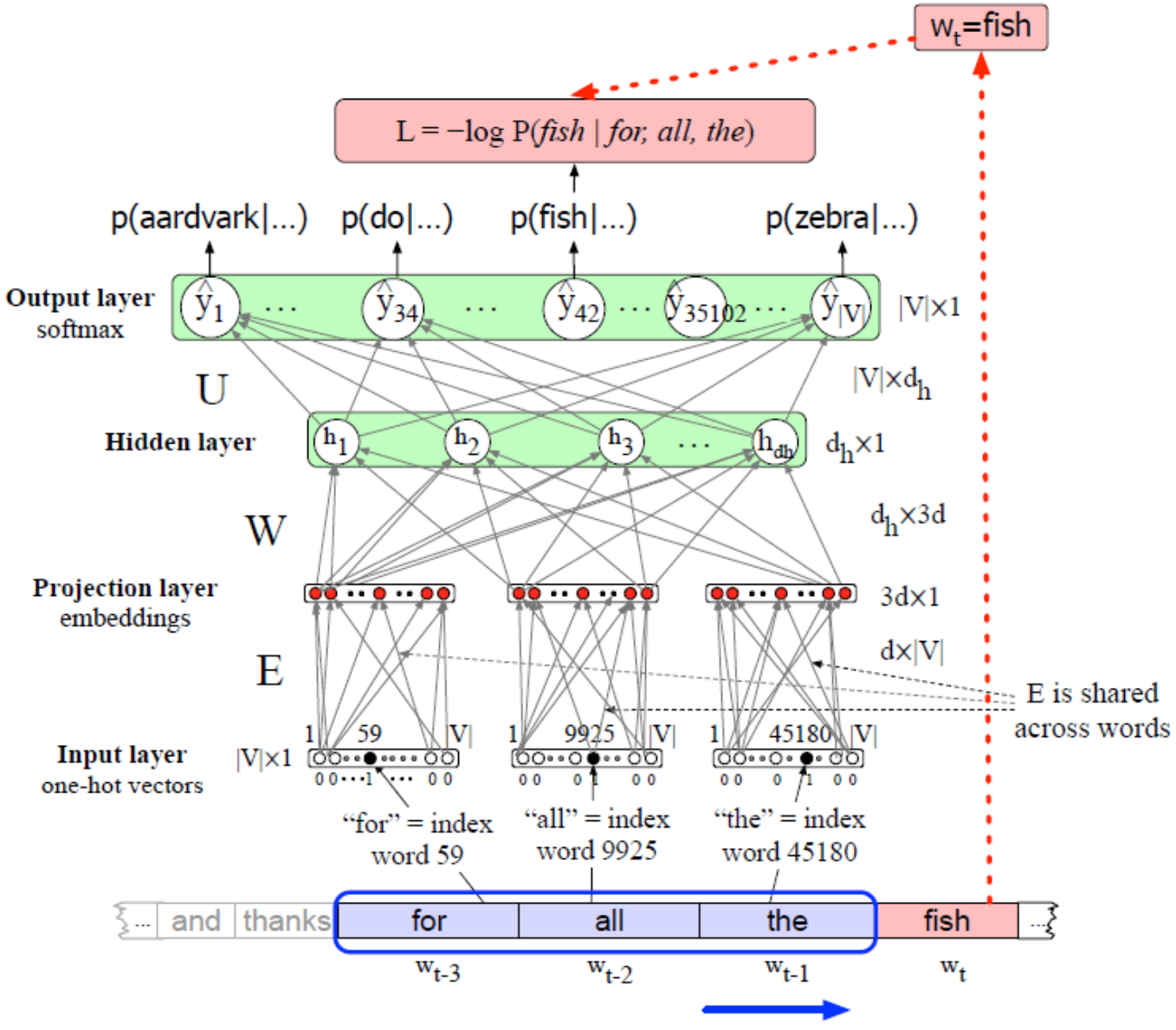
Capacidad de generalización con embeddings (II)

- Primero asumiremos que ya tenemos un diccionario E que nos da, para cada palabra de nuestro vocabulario V, la incrustación de esa palabra.


p(aardvark|...)
p(fish|...)
p(for|...)
p(zebra|...)
Output layer
softmax
$\hat{y}_1$ ... $\hat{y}_{42}$ ... $\hat{y}_{59}$ ... $\hat{y}_{35102}$ ... $\hat{y}_{|V|}$
|V|×1
U
|V|×d_h
Hidden layer
h_1 h_2 h_3 ... h_{dh}
d_h×1
W
w_{t-1}
d_h×3d
Projection layer
embeddings
3d×1
E
embedding for word 35
embedding for word 9925
embedding for word 45180
... and thanks for all the ? ...
w_{t-3} w_{t-2} w_{t-1} w_t

Y si queremos aprender los embeddings?

- En que casos queremos esto?
- Agregaremos una capa adicional a la red y propagaremos el error a esta capa que representará los embeddings.


w_t=fish
L = −log P(*fish* | *for, all, the*)
p(aardvark|...)
p(do|...)
p(fish|...)
p(zebra|...)
Output layer
softmax
y_1
y_{34}
y_{42}
y_{35102}
$y_{|V|}$
|V|×1
U
|V|×d_h
Hidden layer
h_1
h_2
h_3
h_{dh}
d_h×1
W
d_h×3d
Projection layer
embeddings
3d×1
E
d×|V|
E is shared
across words
Input layer
one-hot vectors
|V|×1
1
59
|V|
9925
45180
"for" = index
word 59
"all" = index
word 9925
"the" = index
word 45180
... and thanks
for
all
the
fish
w_{t-3}
w_{t-2}
w_{t-1}
w_t

Neural Language Model

- $e = (Ex_1, Ex_2, \ldots, Ex_{N-1})$
- $h = \sigma(We + b)$
- $z = Uh$
- $\hat{y} = softmax(z)$

Paréntesis para introducir los embeddings

BOW como método de representación

Bow falla en preservar el orden, sin mencionar la alta dimensionalidad. Por ejemplo:

La comida estuvo bien, nada mal.

La comida estuvo mal, nada bien.

Mismo vector representa ambas palabras.

Vector de muy altas dimensiones.

Semántica distribucional

- *'You can tell a word by the company it keeps' Firth 1957*
- *'Distributional statements can cover all of the material of a language without requiring support from other types of information' Harris 1954*
- '***The meaning of a word is its use in the language***', *Wittgenstein 1953*
- *'The complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously.'*

Uso en el lenguaje

las palabras se definen por su entorno (las palabras que las rodean)

Zellig Harris (1954): *Si A y B tienen casi idénticos entornos decimos que son sinónimos.*

¿Qué significa la palabra ongchoi?

Suponga que conoce las siguientes frases:

- *El ongchoi es delicioso salteado con ajo.*
- *Ongchoi es excelente con arroz*
- *Hojas de ongchoi con salsas saladas*

Y también has visto estos:

- *…espinacas salteadas con ajo sobre arroz*
- *Los tallos y hojas de acelga son deliciosos*
- *Col rizada y otras verduras de hojas verdes saladas*

¿Qué significa la palabra ongchoi?

Conclusión:

Ongchoi es una verdura verde frondosa como la espinaca, la acelga o la col rizada.

Similaridad distribucional

Cada palabra = un vector

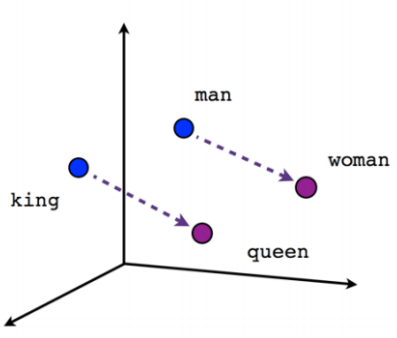
Palabras similares = palabras con el mismo entorno/contexto

Palabras similares deberían estar cerca en el espacio vectorial resultante.

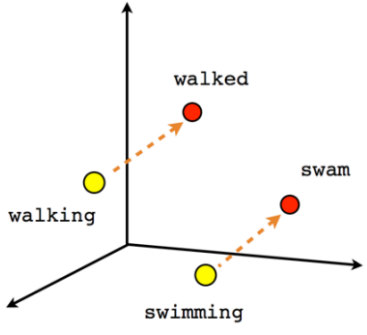
Definimos las palabras como vectores y a esos vectores se le dio el nombre de ***embeddings***

Embeddings: Word2Vec, Glove

Espacios de características más pequeños preservando "en lo posible" el relacionamiento de las palabras.



Male-Female



Verb tense

200-500 dimensiones

Imagen tomada de https://medium.com/@khulasaandh

Gracias por la atención

¿Tiene alguna pregunta?