Agenda

- **anuncios varios**
  - Tarea 2 entrega lunes 8 de Mayo (Preguntas)
- **modelos de analitica (machine learning-ML) Supervisado**
  - **Árboles:**
    - **GBM**
    - **Random forest**
  - **Redes Neuronales**
    - **NNets**
    - **Convolutions**
    - **LSTM**

Supervisado, Árboles

Árboles: Es un algoritmo que parte el espacio de datos para hacer una clasificación o regresión utilizando métricas como gain, entropy.

$$Gain_{(T,X)} = H_{(T)} - H_{(T,X)}$$

Random forest: Utilizar muchos árboles y promediarlos generando muchas muestras con técnicas estadísticas como bagging. Tomando una porción de los datos y de los features

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

GBM Gradient Boosting Machines: Utilizar muchos árboles en serie, entrenar cada árbol utilizando los residuos o datos con errores del árbol anterior.

$$\hat{f}(x) = \sum_{b=1}^{B}\lambda\hat{f}^{b}(x).$$

* A Course of Machine Learning http://ciml.info/

Redes Neuronales

Redes Neuronales: Es un conjunto de combinaciones lineales de los features con una capa de activación no lineal. De esa forma cada neurona aprende combinaciones diferentes que pueden ser combinadas nuevamente con una segunda capa. Dos capas pueden aprender cualquier función continua.

$$\begin{aligned} f(X) &= \beta_0 + \textstyle\sum_{k=1}^{K} \beta_k h_k(X) \\ &= \beta_0 + \textstyle\sum_{k=1}^{K} \beta_k g(w_{k0} + \textstyle\sum_{j=1}^{p} w_{kj} X_j). \end{aligned}$$

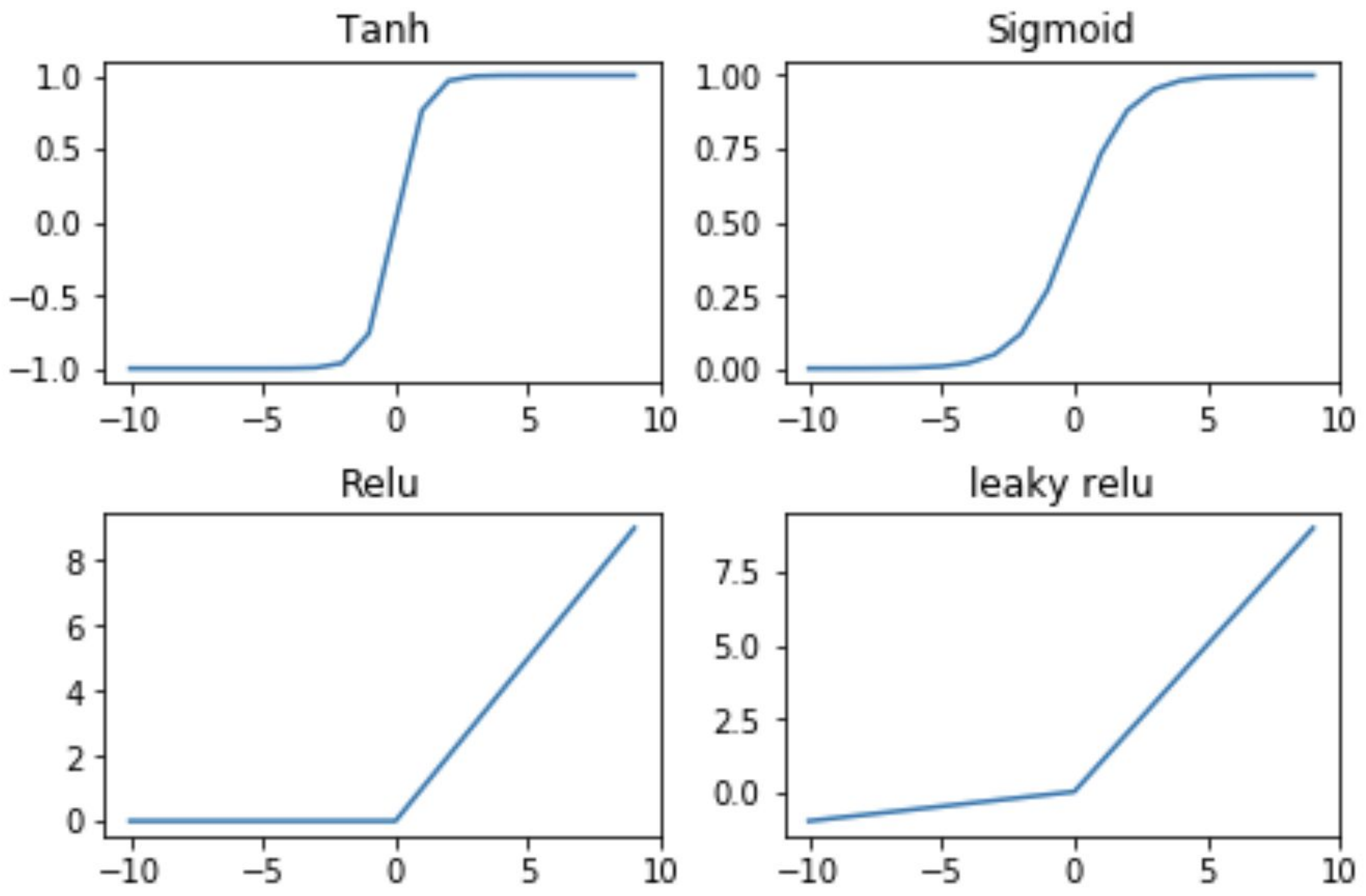
$$A_k = h_k(X) = g(w_{k0} + \textstyle\sum_{j=1}^{p} w_{kj} X_j),$$

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k,$$

K es el numero de Neuronas por cada capa

* A Course of Machine Learning http://ciml.info/

Redes Neuronales, ejemplos de funciones no lineales



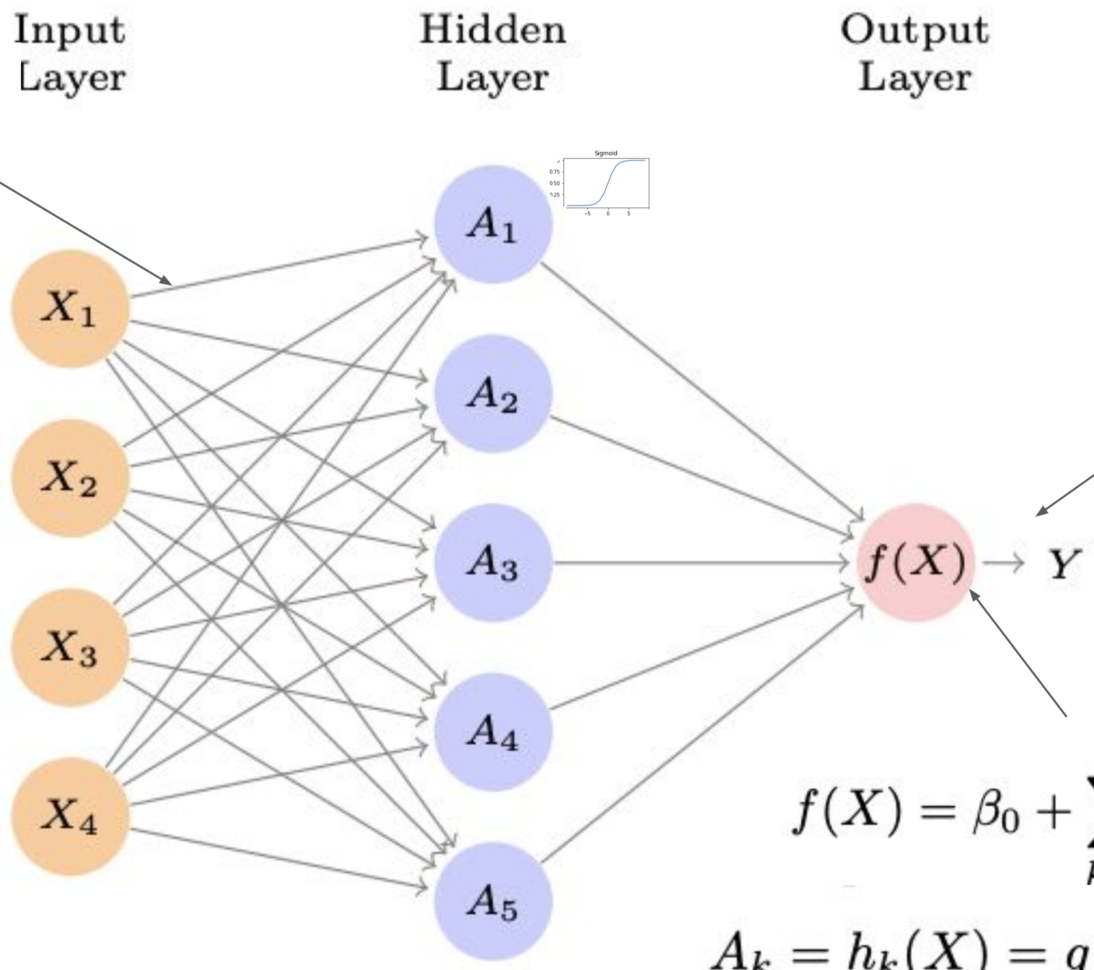
```
fn_tanh = lambda x : ( math.exp(x) - math.exp(-x))/( math.exp(x) + math.exp(-x))
fn_sigmoid = lambda x : 1/(1+ math.exp(-x))
fn_relu = lambda x : max(0,x)
fn_leaky_relu = lambda x : max(0.1*x,x)
```

* A Course of Machine Learning http://ciml.info/

Redes Neuronales

Tomamos los datos iniciales y los multiplicamos por combinaciones aleatorias, aplicamos las funciones no lineales y agregamos, medimos el error y calculamos los gradientes hacia atrás.

$$\frac{\partial(error)}{\partial(w)} = \frac{\partial(error)}{\partial(a)} * \frac{\partial(a)}{\partial(w)}$$



Si es una regresión, nuestro costo o error:

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 .$$

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k,$$

$$A_k = h_k(X) = g(w_{k0} + \textstyle\sum_{j=1}^{p} w_{kj} X_j),$$

* A Course of Machine Learning http://ciml

Redes Neuronales, Backpropagation

Let us simplify and set the bias values to zero, and treat the vectors as scalars, to make the calculus more concise. This means our network has two parameters to train,

$$W^{(1)}$$

and

$$W^{(2)}$$

.

Then the output of the first hidden layer is:

$$h^{(1)} = g^{(1)}(W^{(1)}x)$$

The output of the second hidden layer is:

$$h^{(2)} = g^{(2)}(W^{(2)}h^{(1)})$$

The output of the final layer is:

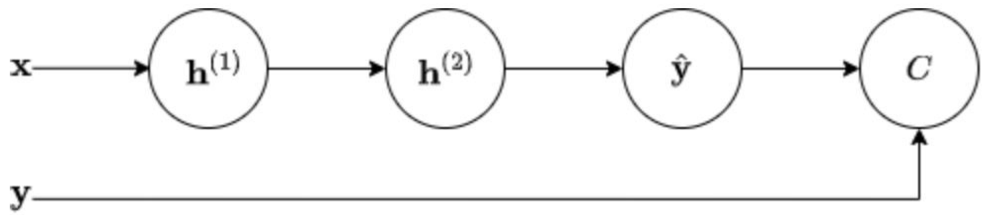
$$\hat{y} = W^{(3)}h^{(2)}$$

And finally, let us choose the simple mean squared error function as our loss function:

$$C(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

and let us set the activation functions in both hidden layers to the sigmoid function:

$$g^{(1)}(x) = g^{(2)}(x) = S(x) = \frac{1}{1 + e^{-x}}$$


$$\begin{tikzcd} \mathbf{x} \arrow[r] & \mathbf{h}^{(1)} \arrow[r] & \mathbf{h}^{(2)} \arrow[r] & \hat{\mathbf{y}} \arrow[r] & C \\ \mathbf{y} \arrow[rrrru] & & & & \end{tikzcd}$$

https://deepai.org/machine-learning-glossary-and-terms/backpropagation

* A Course of Machine Learning http://ciml.info/

Redes Neuronales, Backpropagation Step 1 calculate gradients

It will be useful to know in advance the derivatives of the loss function and the activation (sigmoid) function:

$$\frac{dC}{d\hat{y}} = \hat{y} - y$$
$$\frac{dS}{dx} = S(x)(1 - S(x))$$

Using backpropagation, we will first calculate

$$\frac{\partial C}{\partial W^{(3)}}$$

, then

$$\frac{\partial C}{\partial W^{(2)}}$$

, and then

$$\frac{\partial C}{\partial W^{(1)}}$$

, working backwards through the network.

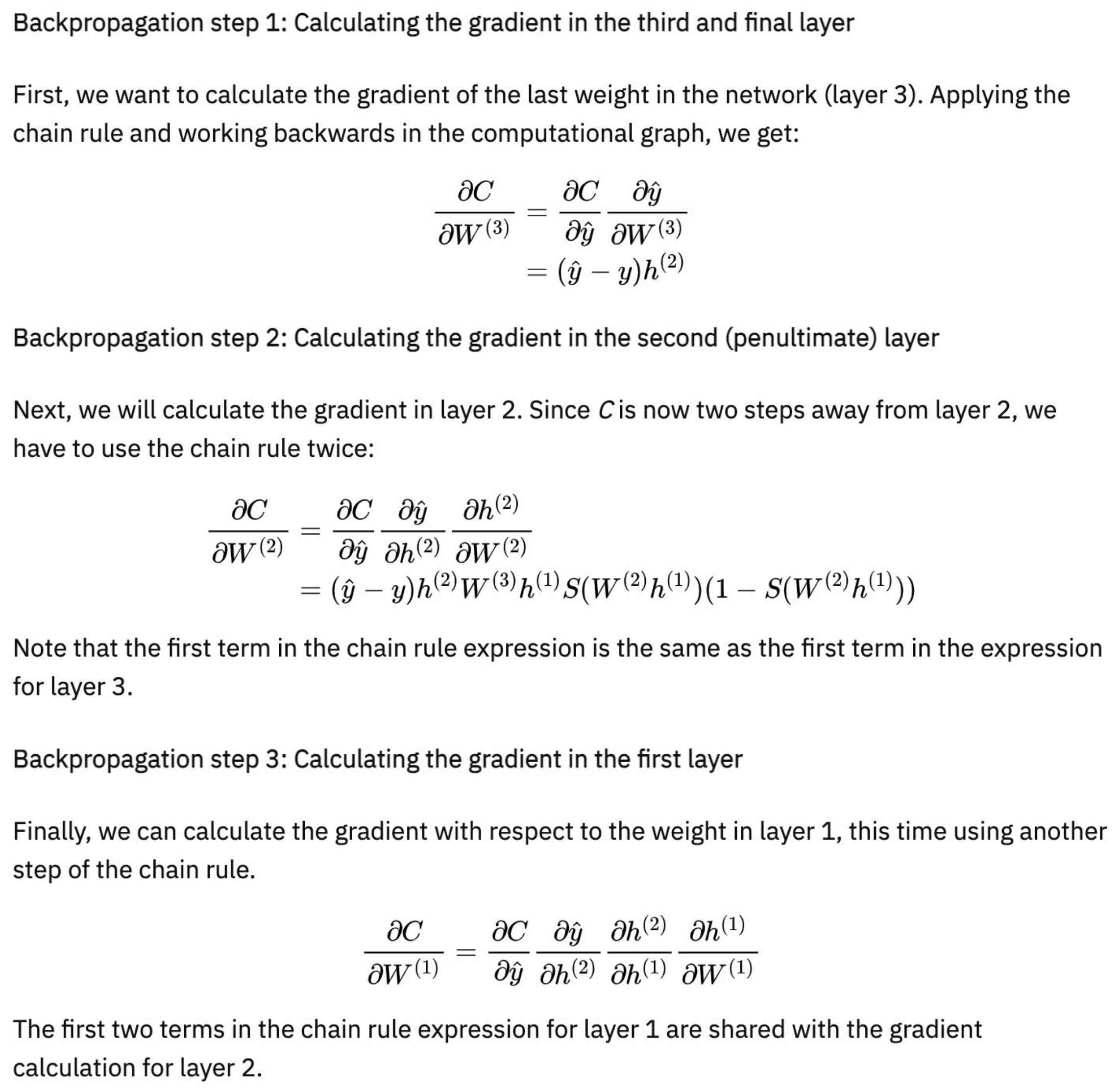
We can express the loss function explicitly as a function of all the weights in the network by substituting in the expression for each layer:

$$\begin{aligned} C(y, \hat{y}) &= C(y, W^{(3)} h^{(2)}) \\ &= C(y, W^{(3)} g^{(2)}(W^{(2)} h^{(1)})) \\ &= C(y, W^{(3)} g^{(2)}(W^{(2)} g^{(1)}(W^{(1)} x))) \end{aligned}$$

https://deepai.org/machine-learning-glossary-and-terms/backpropagation

* A Course of Machine Learning http://ciml.info/

Redes Neuronales, Backpropagation Step 2 calculate error



Backpropagation step 1: Calculating the gradient in the third and final layer

First, we want to calculate the gradient of the last weight in the network (layer 3). Applying the chain rule and working backwards in the computational graph, we get:

$$\begin{aligned}\frac{\partial C}{\partial W^{(3)}} &= \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial W^{(3)}} \\ &= (\hat{y}-y)h^{(2)}\end{aligned}$$

Backpropagation step 2: Calculating the gradient in the second (penultimate) layer

Next, we will calculate the gradient in layer 2. Since C is now two steps away from layer 2, we have to use the chain rule twice:

$$\begin{aligned}\frac{\partial C}{\partial W^{(2)}} &= \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h^{(2)}}\frac{\partial h^{(2)}}{\partial W^{(2)}} \\ &= (\hat{y}-y)h^{(2)}W^{(3)}h^{(1)}S(W^{(2)}h^{(1)})(1-S(W^{(2)}h^{(1)}))\end{aligned}$$

Note that the first term in the chain rule expression is the same as the first term in the expression for layer 3.

Backpropagation step 3: Calculating the gradient in the first layer

Finally, we can calculate the gradient with respect to the weight in layer 1, this time using another step of the chain rule.

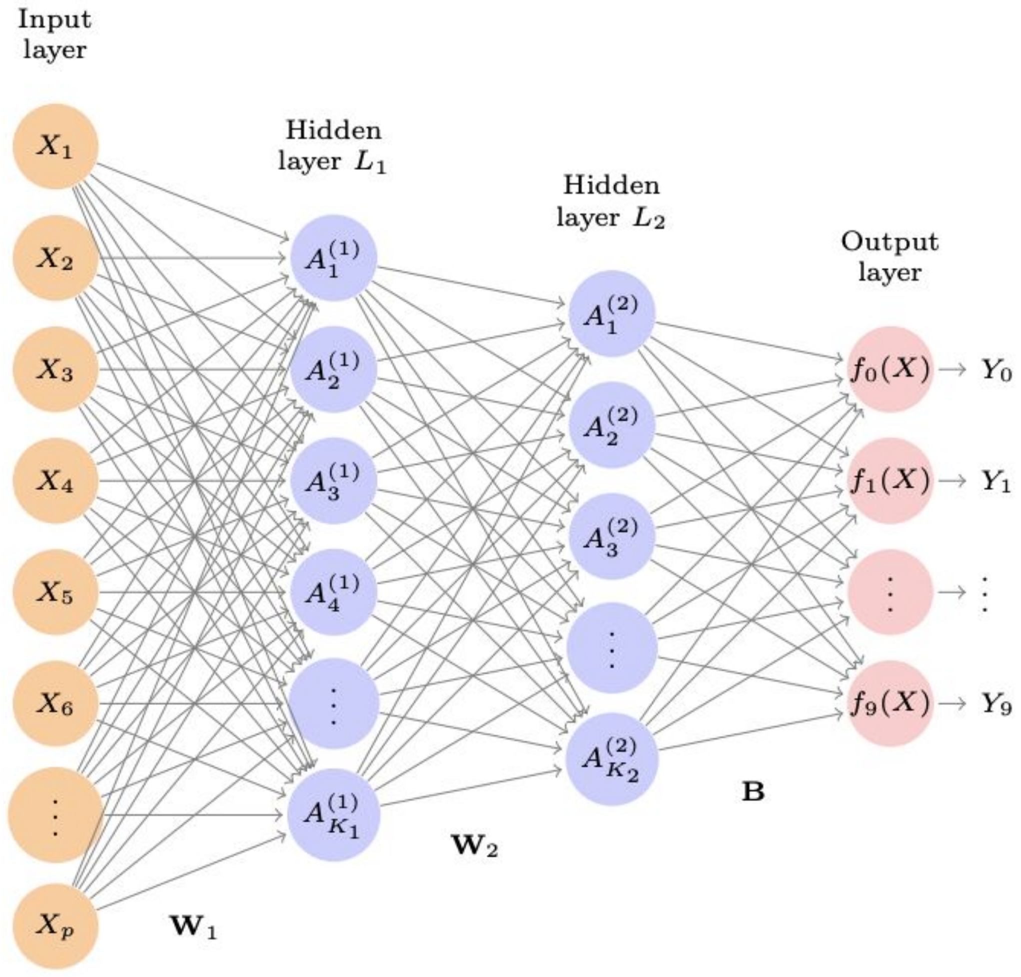
$$\frac{\partial C}{\partial W^{(1)}} = \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h^{(2)}}\frac{\partial h^{(2)}}{\partial h^{(1)}}\frac{\partial h^{(1)}}{\partial W^{(1)}}$$

The first two terms in the chain rule expression for layer 1 are shared with the gradient calculation for layer 2.

https://deepai.org/machine-learning-glossary-and-terms/backpropagation

* A Course of Machine Learning http://ciml.info/

Redes Neuronales, con múltiples capas o layers



[0,0,0,0,0,0,0,0,0,1]

$$-\sum_{i=1}^{n}\sum_{m=0}^{9} y_{im}\log(f_m(x_i)),$$

$$f_m(X) = \Pr(Y = m|X) = \frac{e^{Z_m}}{\sum_{\ell=0}^{9} e^{Z_\ell}},$$

* A Course of Machine Learning http://ciml.info/