**Back Propagation**

Back Propagation determina o erro na saída e propagar de voltar para a rede neural inteira.

O primeiro passo em minimizar o erro é determinar o gradiente de cada neurônio. Então, é uma rede de múltiplas camadas, que determina o gradiente e não é muito straightfoward.

Processos:

1. Escolher uma **arquitetura da rede**, isso é, quantidade de camadas escondidas, quantidade de neurônios em cada camada e função de ativação
2. **Inicializar os pesos** randomicamente
3. Usar **forward propagation** para determinar o neurônio de saída
4. Encontrar o **erro** do modelo
5. **Propagar de volta** o erro na rede neural e determinar o erro em cada neurônio
6. **Atualizar** os **pesos** para minimizar o gradiente

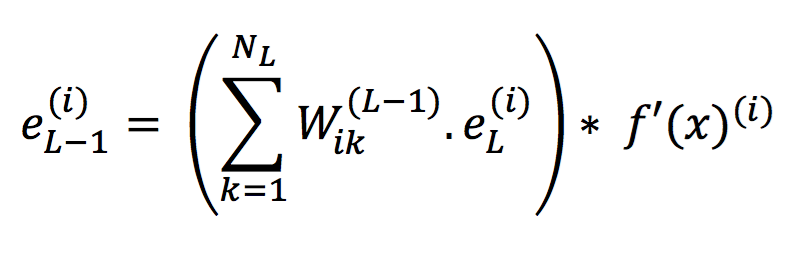
**4 – Encontrando o Erro**

eL(i) = y(i) - aL(i) | i = 1,2,....,NL

y(i) é a saída atual dos dados sendo treinados na rede.

**5 – Back-propogating**

O erro para a camada L-1 pode ser determinado conforme a fórmula abaixo:



Onde i = 0,1,2...NL-1 (quantidade de neurônios na camada anterior (L-1th))

Pontos importantes:

* **Um gradiente de um neurônio é uma função dos gradientes de todos os neurônios da próxima camada.** O erro de um neurônio é baseado nos pesos somados de todos os neurônios da próxima camada onde a saída do seu neurônio vira a sua entrada.
* **f'(x)(i)** se refere à **derivada da função de ativação** para entradas do neurônio. Note que x se refere aos pesos somados de todas as entradas no neurônio presente antes da aplicação da função de ativação.
* **A regra de corrente** é feita pela multiplicação do gradiente do neurônio corrente