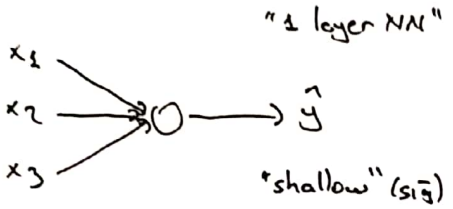
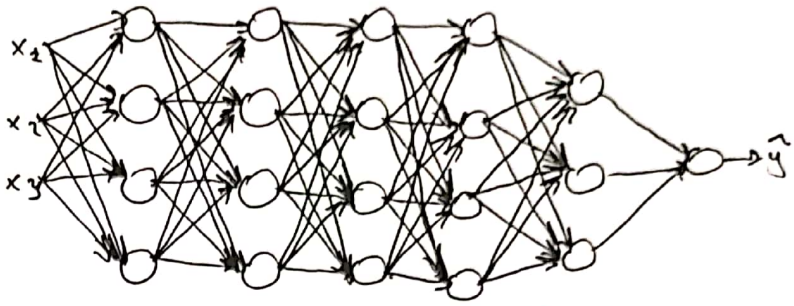
Deep Neural Networks



logistic regression



5 hidden layers "deep"

Bir sinir ağı ne kadar çok gizli katman ile eğitilirse derinliği o kadar artar. Ne kadar az katmanla eğitilirsede sığlığı (shallow) artar. Derinlik arttıkça karmaşık problemlerdeki karmaşık fonksiyonları ayırt etme yeteneğimiz artar. Katmanların sayısını nasıl seçeceğimiz ilerdeki konularda bahsedilecektir.

Deep Neural Network Notation

Not: Sağ üstteki örnek üzerinden anlatılmıştır.

$L = 5$ # Katman sayısı

$n^{[l]}$ = # Katmanlardaki nöron sayısı $\quad n^{[1]}=4,\ n^{[2]}=4,\ n^{[3]}=4,\ n^{[4]}=4,\ n^{[5]}=3,\ n^{[6]}=1$

$a^{[l]}$ = # l katmanındaki aktivasyon $\quad n^{[0]}=n_x=3$

$a^{[l]} = g^{[l]}(z^{[l]})$, $w^{[l]}$ = weights for $z^{[l]}$ $\quad a^{[L]}=\hat{y}$

$b^{[l]}$ = bias for $z^{[l]}$

Forward Propagation in Deep Network

Genel Formül: $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$

$a^{[l]} = g^{[l]}(z^{[l]})$

Yukarıdaki işlemi tüm katmanlar için gerçekleştireceğimiz için katman sayısı kadar bir for döngüsü dönebilir. Bu işlemi vectorize edemeyiz.

Not: W ve b matrislerinin boyutları $W^{[l]} = (n^{[l]}, n^{[l-1]})$ $\quad b^{[l]} = (n^{[l]}, 1)$ formülleri ile genelleştirilebilir. z ve A matrislerinin boyutu $Z^{[l]}, A^{[l]} : (n^{[l]}, m)$ ile genelleştirilir. Türevleri yine kendilerine eşittir. (boyut)

Derin Ağlar Niçin İyi Çalışıyor?

Derin ağlarda ilk katman basit işlevleri tespit eden (resimdeki kenarları bulma gibi) işlemler yapılır. Daha sonraki katmanlarda bunlar kullanılarak daha karmaşık işlevler öğrenilir.

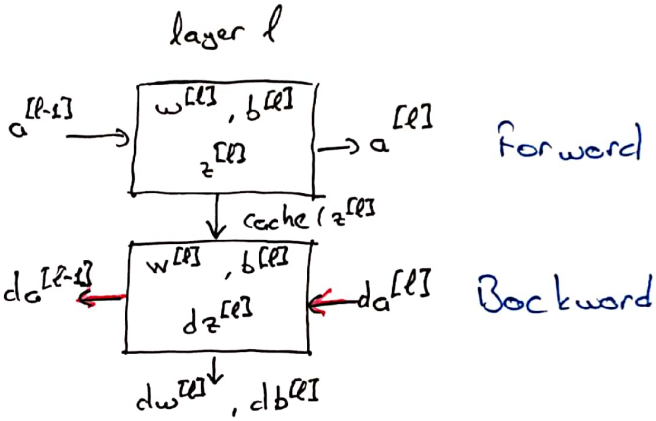
Audio → kısa ses dalgalarını tanıması → fonetik CAT → Kelimeler → Cümle

İnsan beyninde gördüğü bir resimde önce kenarları sonrasında nesneyi tanıyan bir yapıdadır.

Derin ağların çalışıyor gözükmesinin sebebi şu şekilde. Farklı mantıksal durumlarla ne tür fonksiyonlar oluşturabileceğinizi ve hesaplayabileceğinizi düşünen devre teorisinden (circuit theory) gelir.

Kısa ve derin bir sinir ağıyla hesaplayabileceğiniz işlemler var. Bunun yerine daha sığ bir ağla hesaplarsanız e üssü katlanarak daha fazla hidden unit gerekir.

Forward and Backward Functions



Input $a^{[l-1]}$

Output $a^{[l]}$, cache ($z^{[l]}$)

$$z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$
$$A^{[l]} = g^{[l]}(z^{[l]})$$

Input $da^{[l]}$

Output $da^{[l-1]}$, $dw^{[l]}$, $db^{[l]}$

$$dz^{[l]} = dA^{[l]} * g^{[l]\prime}(z^{[l]})$$
$$dW^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1]T}$$
$$db^{[l]} = \frac{1}{m} np.sum(dz^{[l]}, axis=1, keepdims=True)$$
$$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$$

Parametreler ve Hiperparametreler

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \ldots$

Hyperparameters: # learning rate α
iterations (epoch)
hidden layer L
hidden units $n^{[1]}, n^{[2]}, \ldots$
choice of activation function

Hiperparametreler, parametrelerin belirlenmesinde yardımcı olur. Aynı zamanda modelin daha hızlı veya daha verimli bir şekilde eğitilmesini sağlar. Yukarıdakilere ek olarak momentum, mini batch, size, regularization... gibi parametreler de vardır.

Hiperparametreleri deneme-yanılma yöntemi ile belirleriz.