def network(self, weights=None):

model = Sequential() # sequential bir sinir ağı oluşturduk.

model.add(Dense(output\_dim=120, activation='relu', input\_dim=11)) 11 tane input layerımıza özellik koyduk.#Aktivasyon olarak neden reluyu seçtiğimizi asağıda anlattık.

model.add(Dropout(0.15))

model.add(Dense(output\_dim=120, activation='relu'))

model.add(Dropout(0.15))

model.add(Dense(output\_dim=120, activation='relu'))

model.add(Dropout(0.15))

model.add(Dense(output\_dim=3, activation='softmax')) # Son kısım softmax olmalı verileri 0 ile 1 arasına yerleştirir.

opt = Adam(self.learning\_rate) # optimizasyon ekliyoruz

model.compile(loss='mse', optimizer=opt)

if weights:

model.load\_weights(weights)

return model

Aktivasyon Fonksiyonu

Aktivasyon fonksiyonları modele non-linearite katıyor. Gizli katmanlarda (Hidden layer’da) y = f(x,w) şeklindeki lineer fonksiyonumuzda matris çarpımı yapılıp nöronların ağırlığı hesaplandıktan sonra çıktı doğrusal olmayan (non-linear) bir değere dönüştürülür. Çünkü derin öğrenme yöntemleri doğrusal olmayan (non-linear) yapıya sahip problemlerin çözümünde diğer yöntemlere göre daha etkili olduğu için, derin öğrenme yöntemleriyle çözülmeye çalışılan problem genelde doğrusal olmayan non-linear bir problemdir. Matris çarpımı sonucu elde edilen değerin non-linear hale dönüştürülmesi aktivasyon fonksiyonları ile yapılmaktatır. Aktivasyon fonksiyonları çok katmanlı yapay sinir ağlarında doğrusal olmayan (non-linear) dönüşüm işlemleri için kullanılmaktadır. Gizli katmanlarda (Hidden layer’larda) geri türev alınabilmesi (gradient decent hesaplanabilmesi) için (öğrenmede fark geri türevle alınıyor) gizli katmanların (hidden layer) çıktısı bazı aktivasyon fonksiyonları ile normalize edilmektedir. Bu aktivasyon fonksiyonlarından bazıları Şekil 3'de fonksiyon grafikleri verilen sigmoid, tanch, ReLu, PreLu vb’dir. İçlerinde en kullanışlısı ReLu’dur. ReLu’da sigmoid’e göre parametreler daha hızlı bir şekilde öğrenilmektedir. PReLu ise, ReLU’nun kaçırdığı negatif değerleri yakalamaktadır; eğer bizim için negatif değerler önemliyse PReLu tercih edilmelidir.

Aktivasyon fonksiyonlarını gradient descent ile geri dönüp düzeltme yaptığımızda da kullanmaktayız. Bu tarz kullanımdaki amacımız kolay türev alabilmektir.

Şekil 3: En bilinen aktivasyon fonksiyonlarının grafikleri

Aktivasyon fonksiyonlarının bazı özellikleri aşağıdaki gibidir;

İleri beslemeli (Feedforward) ağlarda genelde ReLu versiyonları kullanılmaktadır.

PeRelu, ELU, Maxout’da kullanılması tavsiye edilen diğer aktivasyon fonksiyonlarıdır.

Tanjant hiperbolik de kullanılabilir fakat çok fazla bir şey beklememek gerekiyor.

Sigmoid kullanılması ise genelde tavsiye edilmiyor.

Tanjant hiberbolik sigmoid göre tercih ediliyor.

Google tarafından son dönemde ortaya atılan swish fonksiyonu da çok küçükte olsa başarımı bir miktar artırdığı belirtilmektedir.

Swish, ReLu’ya göre %20 daha yavaş çalışmaktadır. Bununla birlikte 0.001'lik oranında daha başarılı sonuçlar vermektedir.

Epocs ->>> Verinin kaç defa taranacağı

batch\_Size -> Bit genişliği

Aktivasyon Fonksiyonları

Neural networkumuzde nodelar belirli ağırlıklarla çarpılıp toplanır ve yüksek sayılar elde ederiz.Çok katmanlı networklerde bu sayı sonsuza kadar gidebilir. Biz anlamlandırmak ve işlem yapabilmek için verilerimizi 0 ile 1 arasına yerleştiriyoruz.Bunun için de aktivasyon fonksiyonlarının kullanıyoruz.

Relunun kullanılmasının nedeni ise relu sayı/ listedeki max değer “ şeklinde çalışır.Bu yüzden yüksek değerler aynı sonuçları vermez.

Hata Fonksyionları

Gerçek değer – tahmini değer in karesini alıp sonra türveinin alıyor. Duruma göre en iyileme işlemi yapıyor. lossun düşük olması daha iyi

Optimizasyon

Çıkan matematiksel ifadenin türevini alıp learning rate ile çarptığımızda bir önceki nodeun yaklaşık değerini elde ederiz .Learning rate çok yüksek olursa asla istenen değere küçülemez.Çok düşük olursa da overfitting olabilir.

Learning rate ile epoch sayısı arasında ters orantı vardır.

Dropout

Modelimiz eğitim setini ezberlememesi için dropout metodunu kullanıyoruz.Dropout metodu her seferinde bazı rastgele nodelara 0 katsayısı vererek onları iptal eder .Bu sayede modelimiz veri setini ezberlemeiş olur .VAL\_acc 1 e yakın değer olduğunda ezberlemiş olduğunu anlarız. Bu yüzden dropput metodunu uygularız.0.95 bizim için ideal bir sonuç olacaktır.