**Logistic Regression**

* Logistic Regression, Classfication Algoritmalarındandır.
* Genelde 0 ya da 1 sonucu veren datalarda kullanılır.
* Yani Binary Classification içindir.
* Binary Classifcation iki farklı label'ı bulunan datasetlerdir.
* Logistic Regression en küçük neural network'dür.
* Simple Neual Network olarak tanımlanabilir.

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Her bir image 64\*64 pikselden oluşmaktadır. Numpy array’ e dönüştürüldüğünde her bir image 64\*64’den 4096\*1 boyutuna sahiptir.

348 farklı image’i tran ettiğimizde elimizdeki numpy array 4096\*348’lik bir array olur.

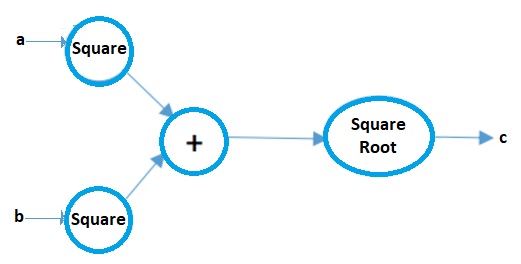
Logistic Regression’u, işaret dilinde ki bir işaretin 0 mı 1 mi olduğunu tanımlamak için kullanabiliriz.

**Computation Graph**

* Matematiksel ifadeleri görselleştirmek için kullanılan bir yöntemdir.
* Logistic Regressionlarda kullanılan matematiksel ifadeleri açıklamak için

computation graphlar yardımcı olarak kullanılabilir.

* Train etmek: ilgili ifadenin kendi modelimize uydurulmasıdır.



**Logistic Regression için Computation Graph**

**metin, harita içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* **Görselde yer alan weightler ve bias parametrelerdir.**
* X\_train kısmı numpy array’de yer alan 64\*64’lük bire image nin bilgilerini vermektedir.
* Weights’ler: coefleri ifade etmektedir. Coefler, independent’dır. Her bir featurenin katsayısını ifade etmektedir.
* Z=(w.t)x + b => Z ifadesine göre pikseller ağırlıkları ile çarpılarak bias ile toplandığında z elde edilir.
* Diğer bir ifadeyle => Z = b + b1x1w1 + bx2w2 + bx3w3 + … + px4096\*w4096
* Z değeri sigmoid Function’ sokulur. Sigmoid Function, activation function olarak ifade edilir.
* Sigmoid function’a girilen değer 0 ila 1 arasında bir sayıya indirgenir, dönüştürülür. Sigmoid function türevi alınabilen bir fonksiyondur. Türevi alınabilmesi weight ve piksel değerlerinin güncellemesine olanak sağlar. Türevi alınamayan fonksiyonlar tek seferliktir. Güncellenemez.
* Sigmoid function’dan çıkan değer “y\_head yani predection”,propability değeri olarak ifade edilir. Propability değer: çıkan değerin 0 ile 1 arasında olmasıdır.
* Sigmoid function’dan çıkan predection değeri bir threshould’a tabi tutulur. Bu threshould değerin 0.9’dan büyük olması durumunda output olarak 1 vermesini söyler.

Bu modelde öğrenilen parametreler **weight** ve **bias** parametreleridir.

Tüm imageleri eğitmek için bias ve weight güncelleme işlemine tabi tutulmalıdır. Ulaşılan yanlış resutların beraberinde bas ve weight güncellenmelidir ki doğru sonuçlar elde edilsin. Her bir image için bias ve weight farklılık gösterir.

Y =a0 + a1\*x ‘den yola çıkarsak a0=bias, a1=weight(kat sayı) ifade etmektedir.

**Initilazating Parameters**

Weight ve Bias, Logistic Regression’a başlanmadan önce belirlenmelidir.

Initilazating value, başlangıç değerini ifade etmektedir. W0,w1,w2….w4096 bu değerler neye göre seçilir bu belirlenmelidir.

Bilinmelidir ki her piksel kendi weightine sahiptir. Bu da 4096 adet weight olduğunu gösterir.

Başlangıç olarak burada weight1 = 0.01 ve bias = 0 değerlerini alacak. Bu değerleri farklı biçimlendirme yöntemleri Artifical neural network kısmında öğrenilecek.

**Forward Propagation**

**Fo**rward propagation: Piksellerdeni cost function’a kadar yapılan işlemlerin tamamına verilen isimdir. Z = w(T)x + b

* Forward Propagation Z ile başlar. Z = piksellerin weightler ile çarpılım bias ile toplanması işlemidir.
* Z’nin sigmaid function’a koyulup propability değer elde edilmesi ile devam eder. Result olarak 0 ila 1 arasında y\_head değeri elde edilir.
* Loss’un bulunması 3. Aşamadır. Yani “1” image’ni predict ettikten sonra 1 result’ına (y\_head = 1) ulaştıysan kayıp hata (loss error)’da 0’dır. Kayıp yoktur. Fakat 1 image’ini predict ettikten sonra “0” result’ına (y\_head =0) ulaştıysan loss(error)=yüksek bir değerdir. Anlaşılacağı üzere loss değerinin minimize tutulması hedeftir.
* Her bir image’için loss function vardır. Her bir loss function’ın toplanması ile cost function elde edilir. Cost function’ın (Maliyet) yüksek olması modelde bir problem olduğunu gösterir, modelin kötü olduğunu gösterir. Cost Function’ın minimum tutulması loss function’a bağlıdır.

**NOT : Loss function 🡺 -y log y\_head - (1 – y) log( 1 – y\_head)**

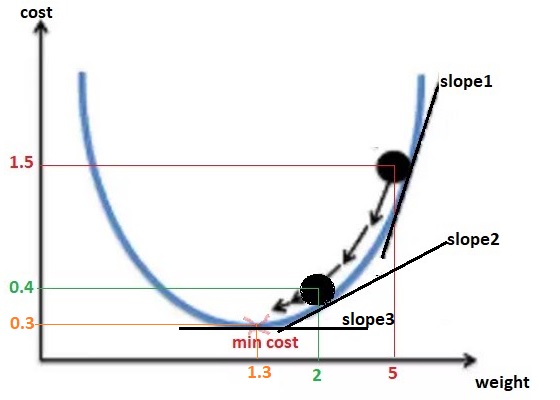
**NOT**: Sigmoid function aşamasından sonra elde edilen y\_head değeri ile y değerinin **kendisi lost function’a sokulur.**  Buradaki y değeri ilk başta ki gerçek değeri, y\_head ise predicted edilen yani tahmin edilen değeri ifade eder. Loss function sonucundan elde edilen değer tutulur ve bu işlem sadece bir image için geçerlidir. **1 image için 1 . forward propagation tamamlanmış olur**. Aynı işlemlerin tüm imagelere yapılması ile image sayısı kadar loss function değeri elde edilir. Aynı işlemlerin 348 image içinde yapılması gerekmektedir. Fakat bu 348 image için bir for döngüsü büyük işlem maliyeti demektir. Bu nedenle numpy array içerisinde sadece 1 forward propagation kalıbı ile 348 image için aynı işlem yapılarak 348 adet loss function değeri elde etmektense 1 adet cost function değeri elde edilir. Loss function değerinin yüksek olmaması gerekir aksi taktirde modelin başarısız olduğunu gösterir. **Bu durumda weight ve bias güncellenmelidir.**

**Optimization Algorithm with Gradient Descent**

**Gradient Descent algoritması ne için kullanılır ? (Dereceli Türevine Göre Azalma Algoritması)**

Oluşturulmuş modelde cost function değerinin yüksek çıkması durumunda modelin kötü bir şekilde oluşturulduğu sonucuna varılır. Cost function değerinden yararlanarak, Gradient Descent algoriması ile weight ve bias değerlerinin optimize edilmesi işlemi yapılır. Kısaca weight ve bias’ın cost’a göre en ideal değerlerin bulunması işlemi için kullanılır.

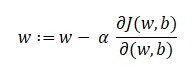
**Tüm forward propagation işlemi tamamlandığında en sondan en başa adım adım dönme işlemine verilen isim ise backward propagation. Bu süreç cost functiondan başlar geriye doğru gider weights ve bias’ı günceller.**



* Unutulmamalıdır ki cost’u 0 değerine ulaştırmak oldukça zor, imkansıza yakın bir işlemdir. Bu nedenle cost’u minimize etmek bizim hedefimizdir.
* Grafikte weight=5 ve bias=0 parametrelerine sahiptir. Bu parametreler sonucunda cost function sonucununda değerinin 1.5 olduğunu anlaşılmaktadır. Kırmızı çizgi ile bu sonuç görülmektedir.
* Weightin azaltılarak güncellenmesi ile cost’un minimuma ulaşması olası durumdur.
* W:=w-step 🡺 w güncellenmesi işlemidir w – step’dir. Step nedir dersek: Step weight’in 5 olduğu ve cost’un 1.5 çıktığı durumda kesişime bakarsak çizilen noktanın diklemesine eğimi bize slope 1’i verir. Slope 1 bizim step’imizdir. Eğimi bulmak için ise ilgili noktanın weight ve bias’a göre türevi alınır.
* Varsayım olarak slope1 yani eğim = 3 değerine sahip olsun. Bu durumda weight=5 🡺 5-3=2 bizim update result’ı yani güncelleme işlemi sonucu yeni weight değerimiz 2 dir.
* Weightin 2 olması ile forward propagation yapılır ve cost 0.4 elde edilmiştir. Cost’u daha da düşürmek için backward propagation işlemi birdaha yapılmalıdır. Yani tekrar weight değeri güncellenmelidir.
* Bunun için w=2, cost=0.4 kesişiminin eğimi(slope 2) alındığında 0.7(step) değeri elde edilir. Bu değerler ile tekrar w:=w-step (güncelleme) yapılırsa 🡺 2 – 0.7 =1.3 new weight değeri elde edilir. Bu grafiğe göre minimum weight değerine ulaşmışızdır.
* Tekrar forward propagation işlemi yapıldığında eğim(slope 3) değeri ise 0.01 (step) olarak bulunmuştur. Bu durumda weight (1.3) - slope3 (step 0.0.1) =1.29~= 1.3 olduğu için minimize cost değeri ve minimize weight değeri elde edilmiştir. Çünkü aynı işlemleri yapmanın artık bir faydası yoktur slope 4 , slope 5… nekadar yapasak yapalım aynı sonuç elde edilecektir, değişim yoktur.

**NOT: Bir grafiğin bir noktaya göre türevini alıp 0’a eşitlenmesi işlemi : ilgili noktanın ilgili grafiğe eğimini bulma işlemidir. Yani slope,slope2…slopen için cost function’un weighte ve biasa göre türevi alarak graf üzerinde geri doğru ilerleriz, slope’nin değeri weightin değeri ile doğru orantılıdır.**

Başka deyişle (w=weight, b=bias, w: = Güncel weight değeri)



Şuana kadar bias’ı, slope’nin hesaplamasında ayrı tutmuştuk çünkü bias’ı default olarak 0 yapmıştık fakat normalde aynı işlemin bias içinde yapılması gerekmektedir. **Yani Cost değerinin hem weight hemde bias’a göre türev alınmalıdır.**

**Diğer bir deyişle: nesne, saat, gök içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Formulde ki “a” değeri “Learning Rate” olarak geçer. Learning rate : Öğrenme hızı, öğrenme oranı olarak değerlendirilebilir.

Learning Rate’in çok düşük veya çok yüksek olması sakıncalı durumdur. Örneklendirecek olursak: A şehrinden B şehrine gidilecektir. Gitme hızı (Öğrenme Hızı) “slow” olursa çok yavaş öğrenme gerçekleşir. ve B şehrine geç varmış olabiliriz, eğer “fast” olursada hiçbir zaman gidemeyebiliriz. Çünkü kaza yapma olasılığımız hız ile doğru orantılı olarak artar. Bu nedenle learning rate’in stabil tutulması gereklidir.

**Learning Rate’in seçilmesi:** Learning Rate, hyperparameter olarak geçmektedir. Weight ve bias parameter’dir. Fakat learning rate hyperparameterdir yani önce seçilir daha sonra deneyerek en doğru hyperparameter elde edilir.

**NOT: Güncel weight ve bias değerlerini bulmak için öncelikle bias ve weighte göre türev alınmalıdır.**

**Daha sonra W:=w – step formülü uygulanmalıdır. Başka bir deyişle:**

**W\_= w – (learning\_rate \* derivative\_weight) #güncel weight değeri**

**B\_ = b – (learning\_rate \* derivative\_bias) #güncel bias değeri**