BLM5110 MAKİNE ÖĞRENMESİ ÖDEV RAPORU

Alican Tunç

Özet

Bu projede, lojistik regresyon yöntemi kullanılarak iş başvurusu yapan kişilerin mülakat ve sınav sonuçlarına göre işe kabul edilip edilmeyeceği tahmin edilen bir model geliştirilmiştir. Veriler, eğitim (%60), doğrulama (%20) ve test (%20) setlerine ayrılmıştır. İlk olarak, veriler görselleştirilmiş ve iki sınıf (işe kabul ve ret) farklı renklerle gösterilmiştir. Modelin eğitimi için sigmoid aktivasyon fonksiyonu ve stokastik gradyan inişi (SGD) kullanılmış, her örnek için ağırlıklar güncellenmiştir. Ayrıca, kayıp fonksiyonu olarak cross entropy loss fonksiyonu tercih edilmiştir.Bu fonksiyon kullanılarak modelin öğrenme süreci izlenmiştir. Eğitim ve doğrulama setlerinin kayıp değerleri arasındaki fark gözlemlenmiş ve bu durumu yorumlarak model geliştirilmiştir. Son parametrelerle belirli bir başarı yakalanmış ve modelin performansı accuracy, precision, recall ve f-score metrikleriyle değerlendirilmiştir.

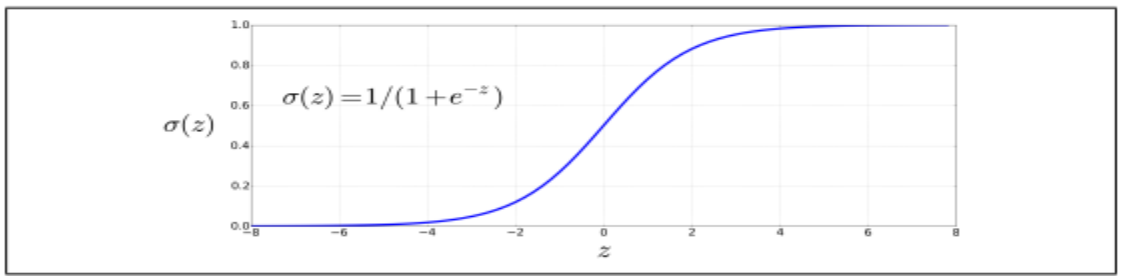
1. GİRİŞ

Öncelikle modelimizi iyi anlamak adına lojistik regresyondan ve kullandığımız metotların teorisinden bahsedeceğiz.Lojistik regresyon, sınıflandırma problemlerinde kullanılan ve özellikle ikili (binary) yanıtları modellemek için tercih edilen bir istatistiksel yöntemdir. Bu yöntem, bir olayın gerçekleşme olasılığını tahmin etmek için kullanılır. Lojistik regresyon, doğrusal regresyondan farklı olarak, çıktıyı sürekli bir değer yerine 0 ile 1 arasında sınırlayan bir model üretir. Bu özellik, modelin özellikle olasılık tahminlerinde doğru sonuçlar vermesini sağlar. Modelin temelinde, doğrusal bir ilişkiyi, sigmoid fonksiyonu gibi bir fonksiyonla ilişkilendirerek, sonuçları olasılık olarak yorumlanabilir bir biçime dönüştürmek bulunur.

Bu kısımda lojistik regresyonun, ikili sınıflandırma problemleri için nasıl kullanıldığı ve bu yöntemin doğrusal regresyon ile karşılaştırılması ele alınmaktadır. Başlangıçta doğrusal regresyon modeli, p(X) = Pr(Y = 1|X) ilişkisini modellemek için kullanılmıştır. Ancak doğrusal regresyon, olasılıkları 0 ile 1 arasında sınırlandırılmadığı için, yanlış sonuçlara (negatif veya 1'den büyük olasılıklar) yol açmaktadır. Bu tür problemleri önlemek için lojistik fonksiyon (sigmoid fonksiyonu) kullanılır. Lojistik regresyon, her X değeri için 0 ile 1 arasında doğru bir olasılık tahmini sağlar.

Bir test örneği üzerinde karar vermek için (eğitimdeki ağırlıkları öğrendikten sonra), sınıflandırıcı önce her bir xi'yi kendi ağırlığı wi ile çarpar, ağırlıklı özelliklerin toplamını alır ve bias terimini b ekler.

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \qquad z = \mathbf{w} \cdot \mathbf{x} + b$$



[1]

Bir olasılık oluşturmak için, z sayısını sigmoid fonksiyonundan geçireceğiz, yani sigmoid(z) Sigmoid fonksiyonu aynı zamanda lojistik fonksiyon olarak da bilinir ve lojistik regresyona ismini verir. Sigmoid fonksiyonu şu şekilde ifade edilir:

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)}$$

[1]

Sigmoid fonksiyonunun birkaç avantajı vardır: Gerçek sayıları alır ve bunları (0,1) aralığına dönüştürür, bu da tam olarak bir olasılık için gereken aralıktır. 0 civarında neredeyse doğrusal, uç noktalara doğru ise düzleşir, bu da uç değerleri 0 veya 1'e sıkıştırmasına neden olur. Ayrıca türevlenebilir olduğu için öğrenme sürecinde faydalıdır. Sigmoid fonksiyonunu ağırlıklı özelliklerin toplamına uyguladığımızda, 0 ile 1 arasında bir sayı elde ederiz. Bunu bir olasılığa dönüştürmek için, p(y = 1) ve p(y = 0) olasılıklarının toplamının 1 olması gerektiğini garanti edebiliriz. Sigmoid fonksiyonu, bir örnek için 0 ve 1 olasılığını hesaplamamıza olanak sağlar. Bu olasılık kullanılarak bir karar verme süreci başlatılır. Eğer P(y=1 | x)P(y = 1 | x)P(y=1 | x) 0.5'ten büyükse, örnek sınıf 1'e (pozitif sınıf) atanır; aksi takdirde sınıf 0'a (negatif sınıf) atanır. 0.5'lik bu eşik, karar sınırı (decision boundary**)** olarak adlandırılır. Bu sınır, örneklerin hangi sınıfa ait olduğunu belirlemek için kullanılır ve genellikle iki sınıfın eşit olasılıkta olduğu noktayı temsil eder.

Lojistik regresyon modelinin başarısını ölçmek için, modelin tahmin ettiği sınıf çıktısı ile gerçek sınıf arasındaki farkı gösteren bir kaybı hesaplamak gerekir. Bu kayıp fonksiyonu, doğru sınıf etiketlerini daha olası hale getirmeyi tercih eder.Yani, modelin çıktısı, modelin doğru etiketini (0 veya 1) maksimize eden ağırlıklar w ve bias öğrenilir. Bu kayıp fonksiyonu, negatif log-olasılık kaybı veya cross-entropy loss olarak bilinir.

Çift sonuçlu (0 veya 1) sınıflandırma problemleri için, her bir gözlem için olasılık hesaplanır ve bu olasılık Bernoulli dağılımı üzerinden ifade edilir. p(y|x) bu olasılık, doğru etiketle modelin tahmin ettiği etiket arasındaki farkı minimizasyon amacıyla logaritması alınarak hesaplanır. Sonuç olarak, cross-entropy loss şu şekilde ifade edilir:

$$\begin{aligned}\log p(y|x) &= \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right] \\ &= y\log\hat{y} + (1-y)\log(1-\hat{y})\end{aligned}$$

[1]

Bu yöntem, doğru sınıfın olasılığını maksimize etmeyi ve yanlış sınıfın olasılığını minimize etmeyi amaçlar. Bu şekilde, cross-entropy loss, modelin ne kadar doğru tahminde bulunduğunu ve hangi noktada iyileştirmeye ihtiyaç duyduğunu belirlemede kullanılır.

Stokastik gradyan inişi (SGD), kayıp fonksiyonunu minimize etmek için her eğitim örneği sonrasında parametreleri (θ) güncelleyerek çalışan bir çevrimiçi algoritmadır. Öğrenme oranı (η), algoritmanın nasıl çalıştığını etkileyen bir hiper parametredir. Çok yüksek olursa aşırı adımlar atılır, çok düşükse işlem uzun sürer. Bu parametre başlangıçta yüksek tutulur ve eğitimle birlikte yavaşça düşürülür. Hiper Parametreleri, modelin öğrenme sürecini etkileyen ve algoritma tasarımcısı tarafından belirlenen parametrelerdir. Bu parametreyi koda aşağıdaki şekilde ekleyebiliriz.

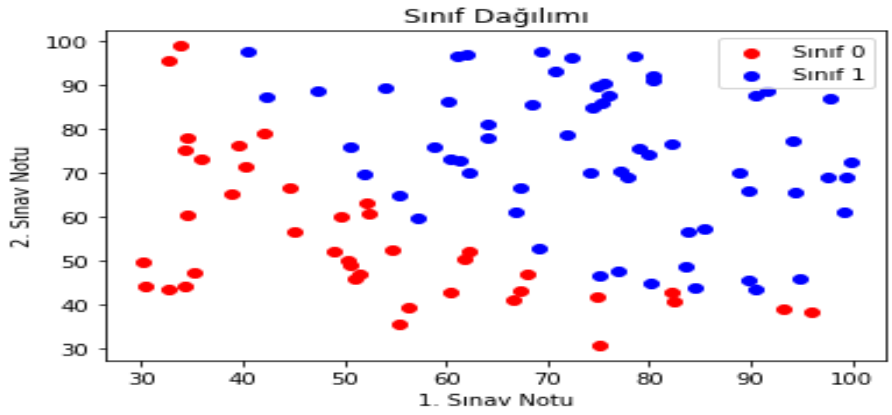
function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ
    # where: L is the loss function
    # f is a function parameterized by θ
    # x is the set of training inputs $x^{(1)}$, $x^{(2)}$, ..., $x^{(m)}$
    # y is the set of training outputs (labels) $y^{(1)}$, $y^{(2)}$, ..., $y^{(m)}$

$\theta \leftarrow 0$ # (or small random values)
repeat til done # see caption
  For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
    1. Optional (for reporting): # How are we doing on this tuple?
      Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}$?
      Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
    2. $g \leftarrow \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
    3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead
return θ

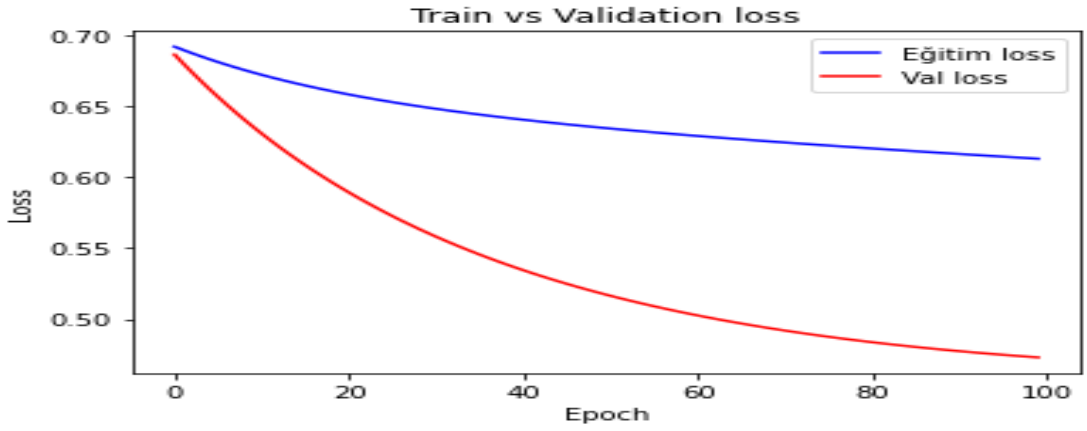
[1]

2.Deneysel Analiz

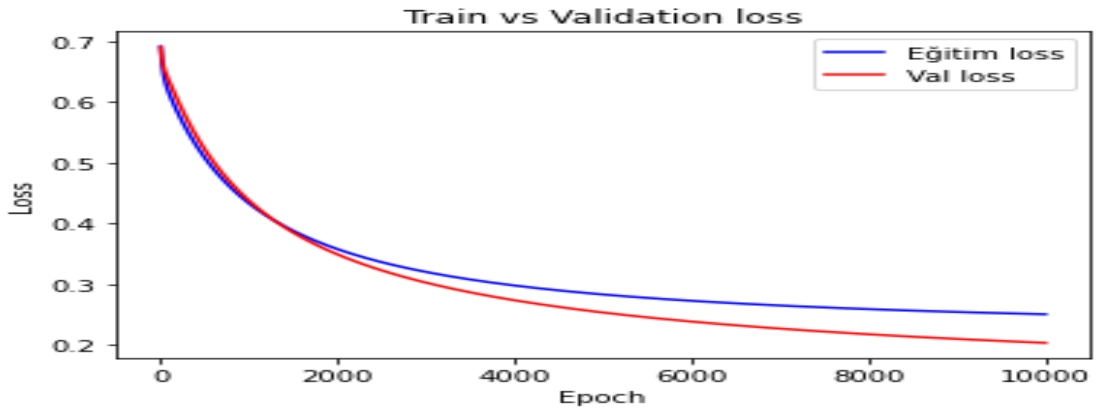
Öncelikle elimizdeki datayı grafiğe döküp şu anki durumda ne olduğunu inceleyelim.



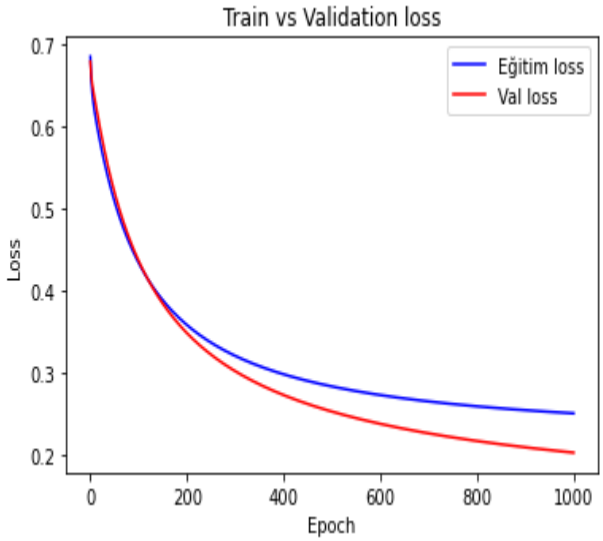
Bu verileri göre elimizde iki adet x1 ve x2 değişkeni ile 0 ve 1'lerden oluşan bir y çıktısı var. Başlangıç için herhangi bir bilmediğimiz için ağırlıkları 0 ve öğrenme oranını çok küçük bir değer verebiliriz. Ayrıca datamızı aynı boyutta olması için normalize edip işlem yapmamız gerekiyor. SGD kullandığımız için zaten tek bir epoch için bile çok fazla ağırlık güncellemesi olacak o yüzden 100 epoch ve 0.01 lr ile başlayabiliriz.

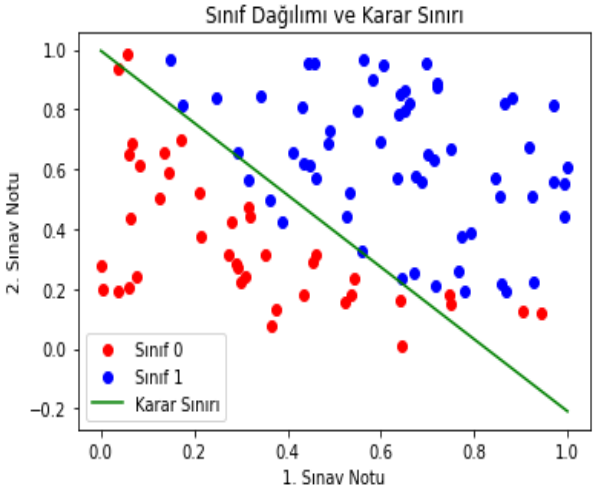


Eğitim kaybı yüksek olduğu için ve yüzde 60 civarı başarı gelmesinden dolayı modelin daha fazla eğitime ihtiyaç duyduğu anlamını çıkarabiliriz. Modeli daha fazla epoch eğitmek veya öğrenme oranını ayarlamak faydalı olabilir.



10000 epoch uyguladığımız zaman grafik buna dönüşüyor. 2000 gelmeden eğitmeyi keser ve lr ile biraz daha oynarsak modelimizi son haline getirebiliriz. 1000 epoch ve 0.01 lr uygulayarak aşağıdaki grafikleri elde ederiz.





3.Sonuç

```
0.9166666666666666 0.9189189189189189 0.9444444444444444 0.9315068493150684
0.95 1.0 0.9166666666666666 0.9565217391304348
0.9 0.8571428571428571 1.0 0.923076923076923
```

Yukarıdaki değerler sırasıyla eğitim, doğrulama ve test için accuracy, precision, recall, f1-score değerlendirmesidir. Beklediğimiz gibi eğitimden sonra doğrulamada başarı artmış ve test sırasında ikisinden daha az olacak şekilde gelmiştir. Zaten özel hazırlanmış bir veri seti olduğu için yukarıdaki gibi karar sınırı çizmemiz basit olmuş ve yüksek başarı oranı gelmiştir.

4.Referanslar

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., draft, Aug. 20, 2024, ch. 5.