**Deep Learning**

Genel olarak basit işlerde DL, diğer ML Modellerden daha iyi sonuç vermez. Bazı şeyleri DL ile yapmalısınız ama ML in yaptığı hemen hemen herşeyi normalde DL ile yapabilirsiniz. Ama zaman ve maliyet dezavantaji oluşturabilir.

**Setting Virtual Envirement**

Virtual Envirement içinde default olarak yalnız Numpy kuruludur. Tensorflow 2.0 dan sonra keras, tensor flow altinda sunuldu. O yüzden Extradan Keras install etmeyeceğiz. Kerasa Tensorflow içinden erişeceğiz. Aksi halde versiyon uyusmazliklari meydana gelebiliyor. Keras import edilerek yazilmis kodlari calistirirken bu kisimlar düzeltilerek calistirilmalidir.

Colabta GPU ile deep learning calistirmak kolay, localde bu is daha zor.

**Neural Network**

Neural Network bir ML Modelidir. İnsan beyni ve sinir sisteminden esinlenerek keşfedilmiş bir modelidir. Neural networklerde, sisteme belli girdiler ve çıkması gereken değerleri veriyoruz. Bu girdiler katsayılar ile çarpılıp toplanıyor. Bu toplamı bir aktivasyon fonksiyonuna sokarak bir çıktı elde ediyoruz. Beklenen değer ile elde ettiğimiz çıktının farkını alarak hata oranını buluyoruz. Ve bu hata oranına göre katsayılarımızı tekrar güncelliyoruz. Ve bu işlem belli bir döngü şeklinde tekrar ediyor. İşte bu kısımda neural networkumüz öğrenme işlemini yapıyor. En nihayetinde katsayılarımız en doğru değerini buluyor.

Neural Network katmanlar şeklinde kurulmuş bir yapıdır.

İlk katman giriş, son katman çıkış olarak adlandırılır.

Orta kısımda bulunan katmanlar ‘Hidden Layers’ adlandırılmaktadır.

Her katman belli sayıda ‘Neuron’ içerir. Bu neuronlar birbirine ‘Synapse’lar ile bağlıdır.

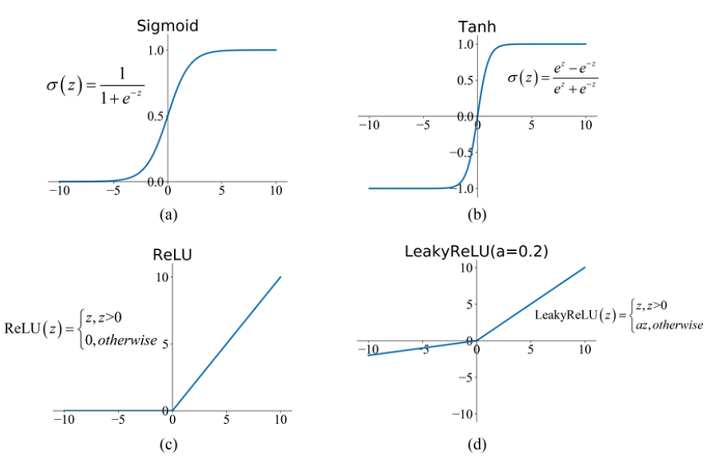
Synapselar bir katsayı barındırır. Bu katsayılar bağlı oldukları neurondaki bilginin ne kadar önemli olduğunu söylemektedir.

**Perceptron**

Perceptron, tek katmanlı neural networktür. Hidden Layer yoktur. Fazla kompleks olmayan sorunların çözülmesinde yardımcı olmaktadır. Neural networklerde asıl istenen katsayıların eğitilmesidir. Katsayılar eğitilirken delta rule kullanılır

**Aktivasyon Fonksiyonu**

Y değeri belirli bir değerin üzerindeyse aktivasyon fonksiyonu activated. Temelde yaptigi is filtrelemedir. Bir kurala bagli olarak cikisi acip kapatan bir step fonksiyonu gibi calisir. çıktı değeri(y) belli bir değer üstünde ise active edilmiş oluyor, oda step fonk.'a bağlı oluyor

****

Temelde 3 aktivasyon fonksiyonu vardir.

* Binary step function
* Linear activation function
* Non Linear activation function

Step function linear bi function. Modern algoritmalar non linear activasyon function kullanıyor. Yani artık yeniler hep active oluyor diyebiliriz. Active olmama durumu eski tip activasyon functionlarda olan bişeydi. relu, softax, sigmoid, tanh hepsi bi sonuç üretiyor.

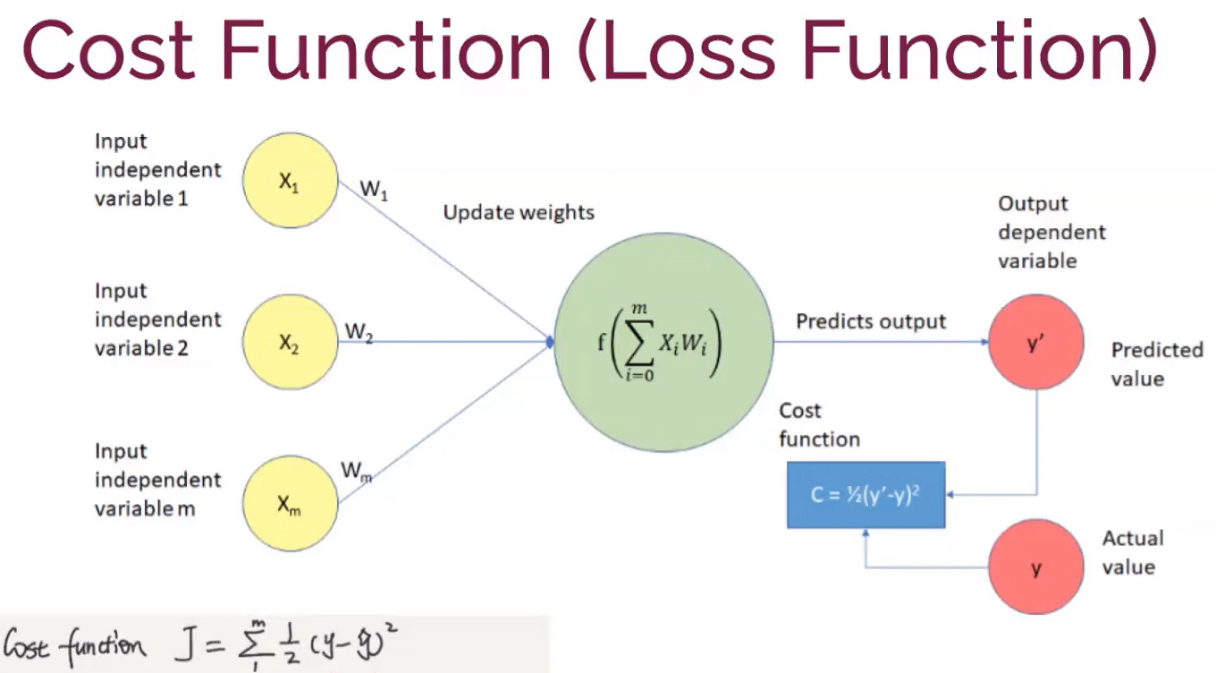
**Network Architectures**

1. **Single-layer feed-forward (Single-layer perceptron):** consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. Aktivation function is sigmoid
2. **Multi-layer feed-forward(Multi-layer perceptron):** Multi-layer networks use a variety of learning techniques, the most popular being back-propagation.
3. **CNN(Convolutional neural network):** Image/video proccessing, video proccessing, recommender systems, medical image analysis, NLP, brain-computer interfaces and financial time series.
4. **RNN(Recurrent neural network /Feedback neural network): For** Time-Series problems. Focus to the prediction of the next step
5. **LSTM (Long short-term memory):** For Time-Series problems. Focus to the prediction of the next step

Every NN has three types of layers: input, hidden, and output.

A neural network is called a deep neural network(deep learning) if it contains two or more hidden layers.

**Cost Function / Loss Function**

****

Cost Function, Tasarlanan modelin hata oranini ayni zamanda basarisini olcer. Modelin yaptigi tahminin hedef değerden ne kadar farkli olduğunu gösterir. Basarisiz modelde bu fark yüksek olacaktır. İyi modelden beklenti 0a yakin olmasidir.

Literaturde Loss ve Cost Function birbiri yerine kullanılabilir. Ancak Loss Function temel, tek bir training sample için hesaplanirken, Cost Function daha genellestirilmis ifade biçimidir ve tum eğitim seti üzerinden bize fikir verir. Cost, tahmin edilen deger ile gercek deger arasindaki farkin karelerinin toplaminin yarisi ile ifade edilir.

Hata fonksiyonuyla feedback yapıyoruz. w katsayilari güncelleyerek cost'u azaltmaya calisiyoruz

Hatayi minimuma düşürecek optimum noktayi bulmaya calisiyoruz. Modelin amaci cost fonksiyonunun değerini minimuma indirmeyi amaçlar. Cost function, değerinin Minimize olduğu yerdeki model parametrelerinin değerini bulmaya yarar.

en son elde edilen y' en son activation değeri a dır. bunun gerçek y ile arasındaki fark olan loss düşürülebilmesi için a nın geriye doğru bileşenlerini ayralrız yani w agırlık katsayıları ve b bias değerleri. tüm fonksiyonlar birbirine bağımlı birinin girişi diğerinin çıkışı olduğundan bu w ve b değerlerini ayrlama işlemi en başa kadar devam eder ve bitince tekrar loss hesaplanır. her bir epoch ta bu işlem tekrar edilerek en iyi sonuc en düşük cost bulunmuş olur.

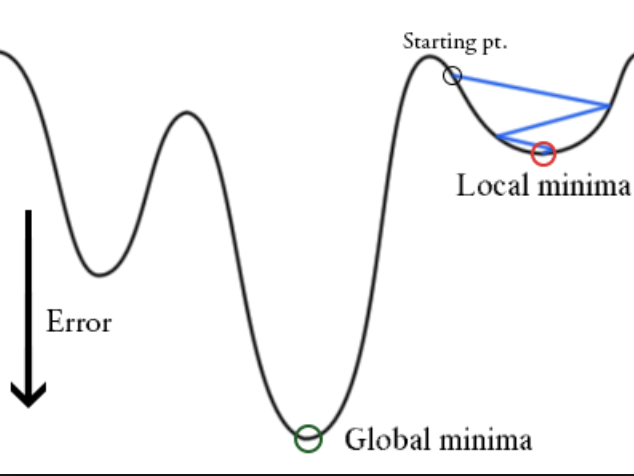
w=w-alpha\*dJ(w,b)/dw

b=b-alpha\*dJ(w,b)/db

j=w ve b parametrelerine bağlı cost function backpropogationda güncelleniyor

Cost Function probleme bagli olarak bircok sekilde olusturulabilmektedir. Cost Function, genelde minimize edilmeye calisilir ancak bazi durumlarda maximize edilmeye de calisilir. Maximizasyonda verdigi deger ödül olarak adlandirilir ve değerinin maximize olduğu yerdeki model parametrelerinin değerini bulmaya calisilir. Minimizasyonda verdigi deger ceza olarak adlandirilir. Her durumda amac optimal model parametrelerini bulmaktir.

Minimum cost function icin: Local minimum ve Global minimum noktalar da olusabilir. Burada global minimum bulmak icin baslama noktasi cok onemlidir. Bunun ayrintilari uygulamada gorulebilir.

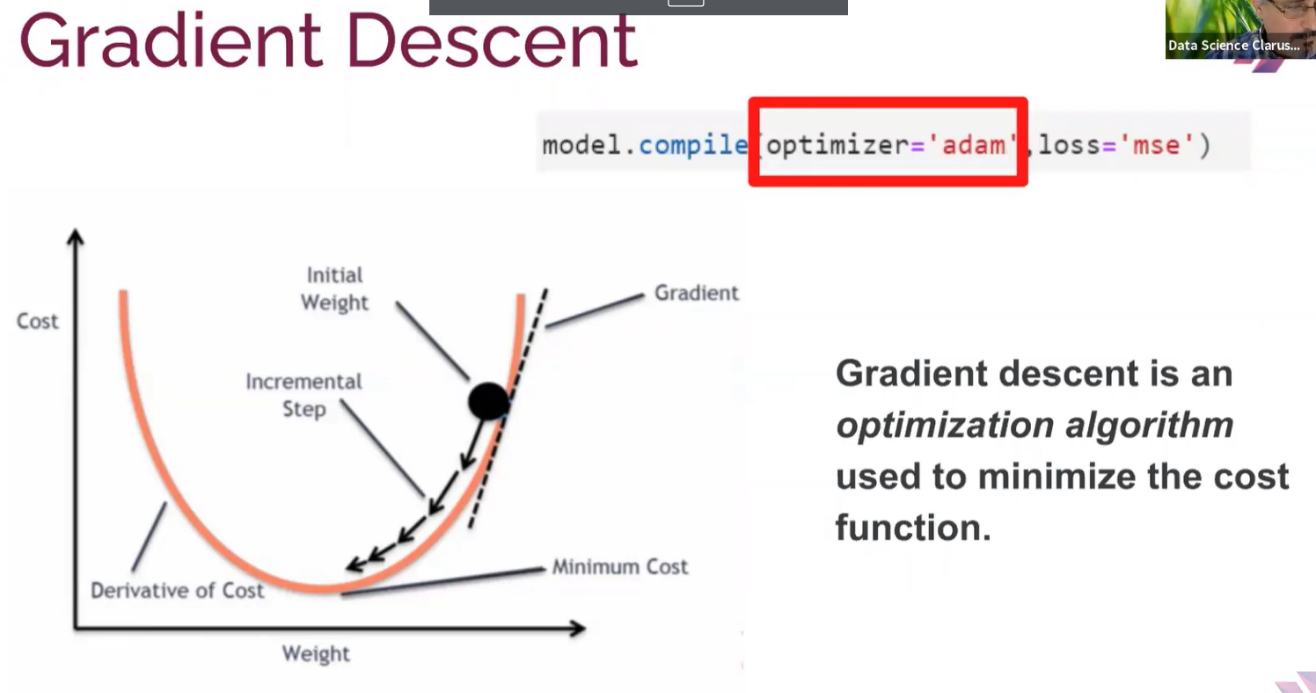


**Gradient Desent**

Gradient Descent, alınan değişkenlerle başlayarak global minimum değerine ulaşmayı amaçlar.

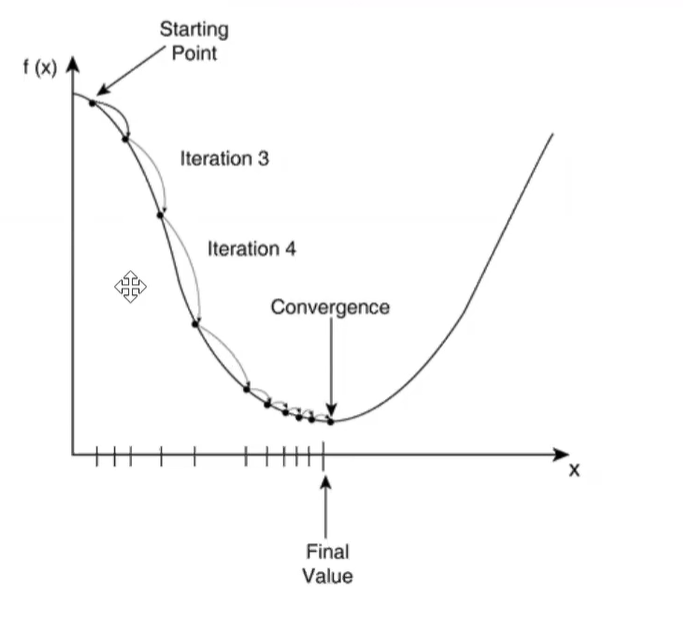
Gradient: Egimin ne kadar dik olduğu

Desent: Asagiya inişi ifade eder.



Turev ile eğime bakilir, learning rate ile minimum noktaya adim adim gidilir.





Gradient Descent adimlari:

1-) Her parametre için loss function in türevini al.

2-) Parametreler için rastgele değerler topla ve toplanan değerleri parametrelere gönder.

3-) step size hesapla.

4-) Yeni parametreyi hesapla.

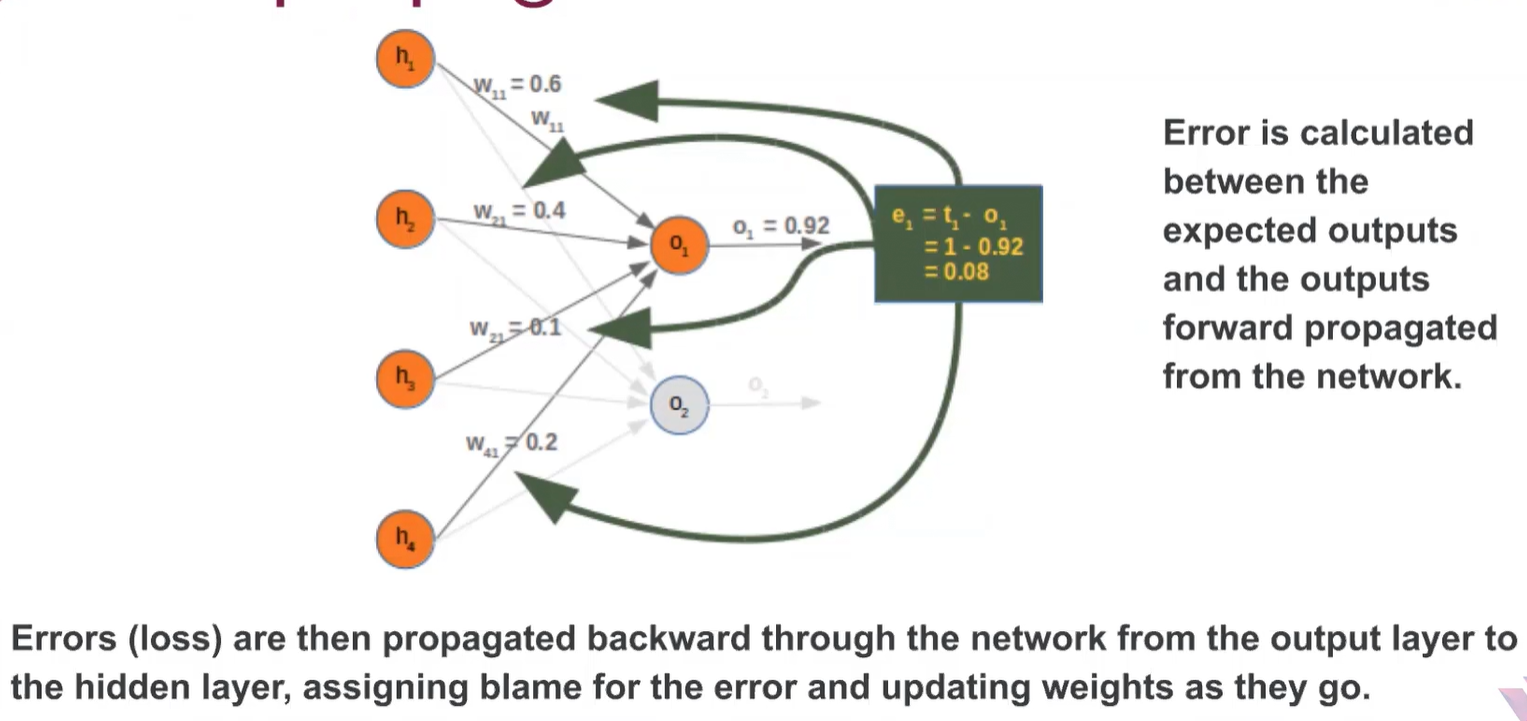
5-) Minimum seviyeye ulaşasıya kadar 2. adıma dön.

The incremental step statement in the figure corresponds to the learning rate concept in deep learning algorithms.

* Gradient Desent bir optimizasyon yöntemidir, bununda farkli cesitleri vardir. AdaGrad, AdaMax, AdaDelta, RMSProd, Adam vs.
* Adam, Gradient Desent i en basarili uygulayan optimizasyon yöntemidir. Starting Pointten minimum cost a giderken buyuk adimlarla baslayip gittikçe küçülen adimlarla devam eder. Boylece arada olabilecek “local minimum” lardan da kurtulunmus olunur. Learning rate yeterince buyuk olmazsa local minimumda gradient sonlanabilir.

**Backpropagation**

Ogrenme denilen olgu aslinda en optimal weight ve biaslerin bulunmasıdır. Hataya bagli olarak geriye donup, weight ve bias leri güncelleme isine backpropagation denir.



Hatalara bir ceza vererek calisir.

It is used algorithm for training feedforward neural networks.

**the objective of a Deep Learning model is to find the right weights and biases that minimize the cost function (error). In this respect, you need to move backward through a network to update the weights and biases. This whole process (going back and updating the weights and biases) is called Backrpropagation.**

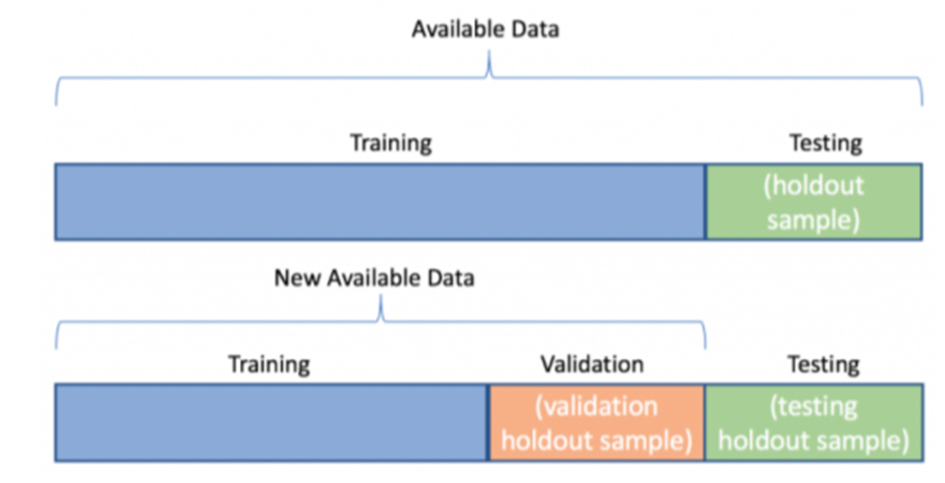
Moving backward through a network to update the weights and biases is called backrpropagation.

the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has learned a certain target function.

Chain ruledan oturu en sondan en basa gidiyoruz.

**Training Dataset – Validation Dataset – Test Dataset**

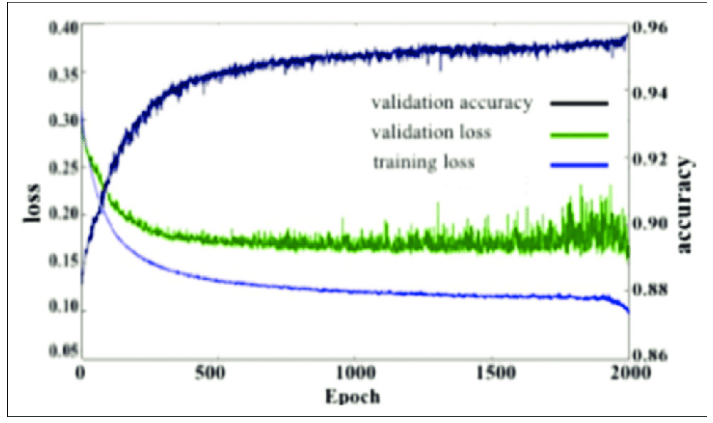
Validation Dataset; Test dataseti ile test etmeden once model, validation dataset ile hizlica test edilir. Eğitilen modeli gogrulamak için yapılır bu islem. Bu ML de test datasetten once train dataset ile CrossValidation yapmak gibidir.



Val\_loss ve val\_acc arasinda birtakim ilişkiler soz konusudur.

1. val\_loss arttikca val\_acc azalmaya başlayabilir. Bu durum öğrenmeyen değerleri sıkıştırdığı anlamina geliyor.
2. val\_loss arttikca val\_acc artmaya başlayabilir. Bu durumda en son katmanda (softmax ile ilgili) overfiting durumu ortaya cikiyor anlamina geliyor.
3. val\_loss azalirken val\_acc artmaya başlayabilir. Bu durum bizim için istenen bir durumdur. Buraya ulaşmaya calisiyoruz.

(Softmax: Maximum likelyhood tabanli olasiliklar üreterek size olasiliklari verir.)



**Epoch**

Training sirasinda Dataset üzerindeki gerçekleştirilen dongu sayisini ifade eder.

**Batch Size**

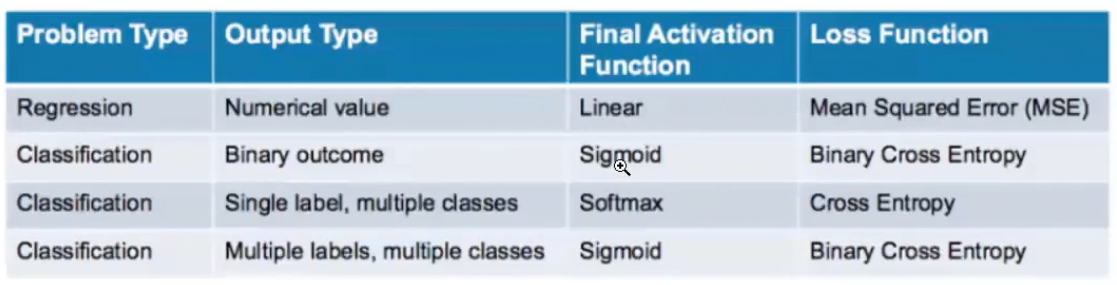
Bir iterasyonda kullanilan train samplelarin sayisini ifade eder. Ram kisitindan dolayi tum dataset ayni anda Neural Network üzerinde iterasyona sokulmaz. Grup sayisi belirlenip ona gore iterasyona sokulur, Batch size ne kadar küçükse eğitim de o kadar uzun sürecektir. Mesela batch\_size:32 verirsek X\_train shape i 700 ise, 700/32= 22 iterasyonda train işlemi tamamlanir.

**Drop-out Technic**

* Modelimizde Ilk once 2-3 layerla sonuç alinir.
* Sonra layer sayisi artirilarak evaluation metrics izlenir. Layer artiriminin accuracy de artik iyileşmeye bir etkisi olmadigi optimum nokta yaklasik tespit edildikten sonra layer sayisi orada sabitlenir.
* Sonra ilk layerdan son layera kadar nöron sayisinin kademeli olarak (bir huni gibi) düşürülerek ayarlanmasi yapılır, Bunu yapmak daha iyi sonuç elde etmemizi sağlar. Ve Regularization etkisi gösterir.
* Drop yapılacak nöronlar, iyi değer vermeyen, ornegin 0.5in altinda accuracy e sahip olanlardan seçilir. Buna **Drop-out Technic** i denir.

**Other Notes:**

* The number of hidden neurons should be between the size of the input layer and the size of the output layer.
* The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
* The number of hidden neurons should be less than twice the size of the input layer
* DL nin outliera sensitivedir. Bu nedenle mutlaka scale edilir.
* Regreyon problemlerinde output layerda 1 noron vardir, aktivasyon fonksiyonu yoktur.
* binary classificationda output layerda nöron sayisi 2 dir. aktivasyon fonk sigmoid tir.
* Multiple classificationda output layerda nöron sayisi idi kac clas varsa o kadardir. aktivasyon fonk softmax tir.
* Input ve hidden layers ta aktivation function olarak genelde relu kullanılır.
* Loss function asagidaki tabloya gore seçilir.



**hidden neurons < 2\* size of input layer**

**hidden neurons ≈ 2/3 \*size of input layer +** **size of output layer**