T.C.

KARADENİZ TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

YAPAY SİNİR AĞLARI PROJE ÖDEVİ

330104

ONUR ERDAŞ

1.ÖĞRETİM

Prof. Dr. MURAT EKİNCİ

Proje Konusu:

Yapay sinir ağları kurarak sınıflandırma yapmak

Proje Amacı:

Yapay sinir ağları temeli öğrenmek amaçlı verilen iki veri kümesini sınıflandıran doğruyu bulmak

İçindekiler:

- Single Layer Discrete Learning
- Single Layer Continuous Learning
- Multicategory Single Layer Discrete Learning
- Multicategory Single Layer Continuous Learning

Proje Çıkarımları:

Ayrık ve sürekli fonksiyonlar arasındaki farkların görülmesi

Eğitim sürecinde güncellenen doğrunun gözle görülmesi

Single Layer Discrete Learning

Burada yapmış olduğum iş verilen iki sınıfı bir doğru ile ayırma işlemidir. Visual Studio'da oluşturduğum CLI C++ projesine bir picturebox ekledim.

Koordinat düzlemi çizdirme kodu aşağıdaki gibidir.

```
e->Graphics->DrawLine(object, Point(0, Center_Y), Point(Size_X, Center_Y));
e->Graphics->DrawLine(object, Point(Center_X, 0), Point(Center_X, Size_Y));
```

Burada iki radiobutton ile seçilen sınıfların noktalarını picturebox_mouseclick ile belirledim. Kodu aşağıdaki gibidir.

```
if (radio_class1->Checked) {//for single category class
                    graphic->DrawEllipse(Pens::Yellow, e->X, e->Y, 3, 3);//draw a yellow
circles for showing the points
                    point->setD(-1);//set D -1 for Class1
                    points.push_back(point);
             }
             else if (radio_class2->Checked) {
                    graphic->DrawEllipse(Pens::White, e->X, e->Y, 3, 3);//draw a white
circles for showing the points
                    point->setD(1);//set D 1 for Class2
                    points.push_back(point);
             }
```

Sonrasında çiz butonuna tıkladığımda bu iki sınıfı birbirinden ayıran doğruyu çizdirdim. Bu sınıfları birbirinden ayırmak için sınıf1 d değerine -1 sınıf2 d değerine ise +1 atadım.

Eğitim sürecinde her bir adımdaki hesaplamalar aşağıdaki gibidir.

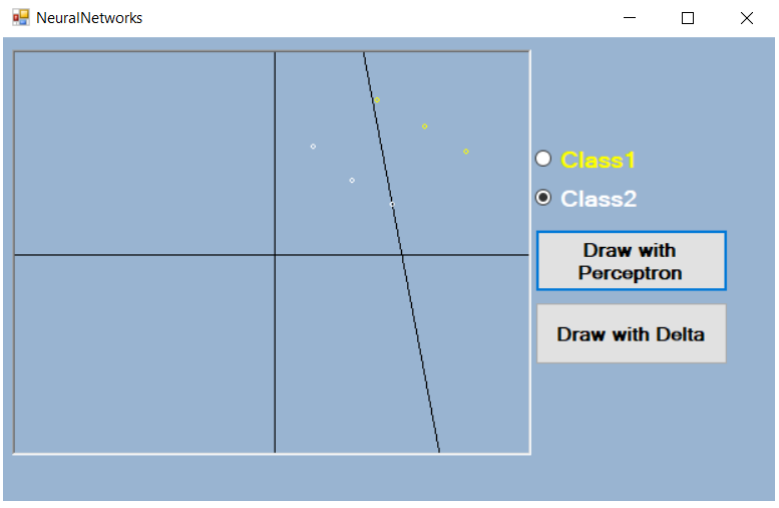
Net değerini doğru değerlerimin transpozu ile bu adımdaki noktanın x,y ve z değerlerini çarpıp hesaplıyorum. Aktivasyon fonksiyonum ayrık bir fonksiyon olduğundan dolayı net değerim sıfırdan büyük bir değer olması durumunda çıkış (sign(net)) değerim 1 ve sıfırdan küçük olması durumunda çıkış (sign(net)) değerim -1 olmakta. Sonrasında danışman değeri(d) ile sign(net) yani çıkış fonksiyonunu karşılaştırıyorum. Bu iki değer birbirine eşit değilse doğru değerlerimi güncelliyorum. Bu güncelleme formülü W(2) = W(1) + c*(d-o)*x ile bulunur. Buradaki c ise öğrenme sabitidir. Bir eğitim döngüsü içerisinde tüm adımlarda çıkış değerleri ile danışman değerleri birbirine eşit ise eğitim burada sona ermektedir. Bu iki sınıfı ayıran bir doğru çizilmektedir. Bu anlatımın kod hali aşağıdaki gibidir.

```
while (points.size() > counter) {

		counter = 0;
		for (int i = 0; i < points.size(); i++) {

			net = line->getA() * points[i]->getX()
				+ line->getB() * points[i]->getY()
				+ line->getC() * points[i]->getZ();

			if (net < 0)
				sgn = -1;
			else
				sgn = 1;
		line->setA(line->getA()
		+ c * (points[i]->getD() - sgn) * points[i]->getX());
		line->setB(line->getB()
		+ c * (points[i]->getD() - sgn) * points[i]->getY());
		line->setC(int(line->getC()
		+ c * (points[i]->getD() - sgn) * points[i]->getZ()));
			if (points[i]->getD() == sgn) {
				counter++;
			}
		}
		pictureBox1->Refresh();
	}
	MessageBox::Show("Line has been drawn successfully!");
```

Doğruyu çizdiren kod aşağıdaki gibidir.

```
int x = (Center_Y - ((line->getA() * (Center_X) - line->getC()) / (line->getB())));
int y = (Center_Y - ((line->getA()) * (-Center_X) - line->getC()) / (line->getB()));
```

Bu kodun çıktısı aşağıdaki gibidir.



Single Layer Continuous Learning

Burada yapılan işlem yukarıdaki gibi verilen iki sınıfı bir doğru ile ayırma işlemidir. Burada almış olduğumuz noktaları (x-x(ortalama))/sapma ile normalize ediyorum. Eğitim sürecinde her bir adımdaki hesaplamalar aşağıdaki gibidir.

Normalizasyon kodu aşağıdaki gibidir.

```
double averageX = 0, averageY = 0, deviationX = 0, deviationY = 0,
       totalX = 0, totalY = 0;
for (int i = 0; i < points.size(); i++) {
       averageX += points[i]->getX();
}
for (int i = 0; i < points.size(); i++) {
       averageY += points[i]->getY();
}
averageX /= points.size();
averageY /= points.size();
for (int i = 0; i < points.size(); i++) {
       totalX += pow(points[i]->getX() - averageX, 2);
}
for (int i = 0; i < points.size(); i++) {
       totalY += pow(points[i]->getY() - averageY, 2);
}
deviationX = totalX / (points.size() - 1);
deviationX = sqrt(deviationX);
deviationY = totalY / (points.size() - 1);
deviationY = sqrt(deviationY);

for (int i = 0; i < points.size(); i++)
{
       points[i]->setX((points[i]->getX() - averageX) / deviationX);
       points[i]->setY((points[i]->getY() - averageY) / deviationY);
}
```

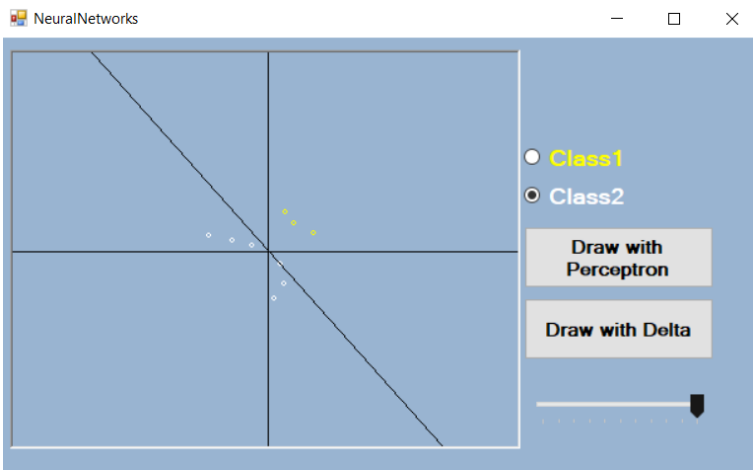
Net değerini doğru değerlerimin transpozu ile bu adımdaki noktanın x,y ve z değerlerini çarpıp hesaplıyorum. F(net) = (2/(1 + e^-(net))) – 1 formülü ile f(net) değerini sonrasında F’(net) = (1-f(net)^2)/2 formülü ile F’(net) değerini hesaplıyorum. Yine burada danışman değeri ile f(net) yani çıkış fonksiyonunu karşılaştırıyorum. F(net) değerim danışmam değerime(d) eşitse doğru değerlerimi güncellemiyor eşit değilse doğru değerlerimi güncelliyorum. Bu güncelleme formülü W(2) = W(1) + c*(d-f(net))*f’(net)*x ile bulunur. Aktivasyon fonksiyonum sürekli bir fonksiyon olduğundan dolayı hata değerim sıfıra yakın bir değer olana kadar bu eğitim işlemim devam eder. Hata değeri belirli bir değerin altına indikten sonra bu iki sınıfı neredeyse tam ortadan ikiye bölen bir doğru çizilmektedir.

Bu anlatımın kodu aşağıdaki gibidir.

```
do {
	err = 0;
	for (int i = 0; i < points.size(); i++) {
		//net calculation
		net = line->getA() * points[i]->getX()
		+ line->getB() * points[i]->getY()
		+ line->getC() * points[i]->getZ();
		long double ex = exp(-net);
		long double a = (2 / (1 + ex));
		//f(net) and f(net)_derivation calculation
		fnet = a - 1;
		fnet_derv = (1 - pow(fnet, 2)) / 2;
		//Update value of w vector
		line->setA(line->getA()
		+ c * (points[i]->getD() - fnet)
		* fnet_derv * points[i]->getX());
		line->setB(line->getB()
		+ c * (points[i]->getD() - fnet)
		* fnet_derv * points[i]->getY());
		line->setC(double(line->getC()
		+ c * (points[i]->getD() - fnet)
		* fnet_derv * points[i]->getZ()));
		err += pow((points[i]->getD() - fnet), 2) / 2;
	}
	pictureBox1->Refresh();//drawing line and points
} while (err > eps);
MessageBox::Show("Line has been drawn successfully!");
```

Eğitim sona erdikten sonra trackbar ile noktaları büyütüp koordinat düzleminde yerlerini görüyorum. Bu işlemin kodu aşağıdaki gibidir.

```
private: System::Void trackBar1_Scroll(System::Object^ sender, System::EventArgs^ e) {
	for (int i = 0; i < points.size(); i++)
	{
		points[i]->setX(points[i]->getX() * 5);
		points[i]->setY(points[i]->getY() * 5);
		points[i]->setZ(points[i]->getZ() * 5);

	}pictureBox1->Refresh();
}
```



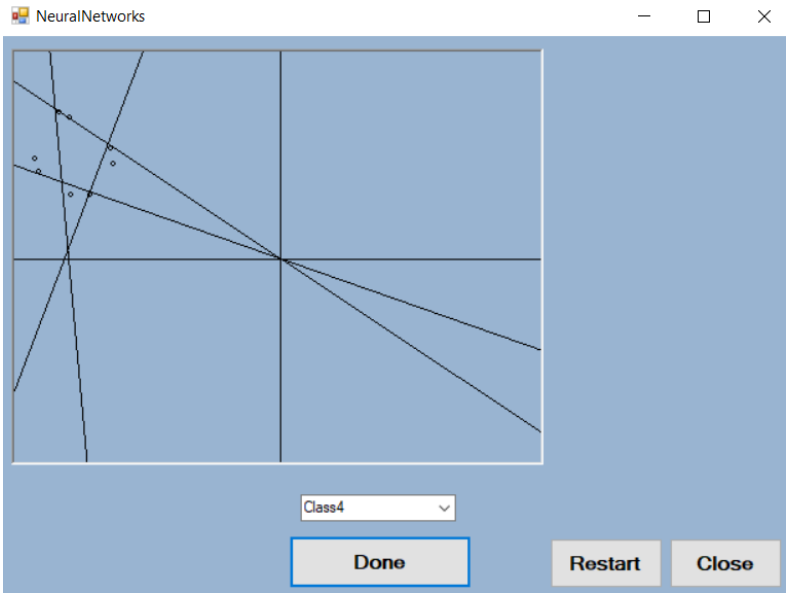
Multicategory Single Layer Discrete Learning

Burada doğru aldığım sınıf sayısına göre bir doğru dizisi oluşturuyorum. Bu kod aşağıdaki gibidir.

```
for (int i = 0; i < numberofclass; i++)//set random values for the linevector and push
them into vector
	{
		Line^ newLine = gcnew Line();
		linevector.push_back(newLine);
	}
```

Buradaki işlemler Single Layer Discrete Learning'deki gibidir. Burada her bir sınıf için ayrı bir doğru değerim vardır. Bu doğruları güncellerken doğru dizi değerlerimi sign(net) dizi değerlerimle karşılaştırmaktayım. Burada eşit olmayan sign(net)[i] ve d[i] değerleri için w[i] doğru dizi değerimi güncellemekteyim. Bu kod aşağıdaki gibidir.

```
int counter = 0, c = 1, control_flag;
	do {
		control_flag = 0;
		counter = 0;
		while (counter < points2.size()) {
			for (int i = 0; i < numberofclass; i++)
			{
				//net calculation
				net[i] = linevector[i]->getA() * points2[counter]->getX()
					+ linevector[i]->getB() * points2[counter]->getY()
					+ linevector[i]->getC() * points2[counter]->getZ();
			}
			for (int i = 0; i < numberofclass; i++)
			{
				//finding value of sign(net)=f(net)
				if (net[i] < 0)
					sgnnet[i] = -1;
				else
					sgnnet[i] = 1;
			}
			for (int i = 0; i < numberofclass; i++)
			{
				if (points2[counter]->getDArray()[i] != sgnnet[i]) {//Update
the value of linevector with respect to output and d value
					linevector[i]->setA(linevector[i]->getA() + c *
(points2[counter]->getDArray()[i] - sgnnet[i]) * points2[counter]->getX());
					linevector[i]->setB(linevector[i]->getB() + c *
(points2[counter]->getDArray()[i] - sgnnet[i]) * points2[counter]->getY());
					linevector[i]->setC(linevector[i]->getC() + c *
(points2[counter]->getDArray()[i] - sgnnet[i]) * points2[counter]->getZ());
					control_flag++;
				}
			}
			counter++;
			pictureBox1->Refresh();
		}

	} while (control_flag > 0);
	MessageBox::Show("Line has been drawn successfully!");
```



Multicategory Single Layer Continuous Learning

Buradaki işlemler Single Layer Continuous Learning'deki gibidir. Burada her bir sınıf için ayrı bir doğru değeri olmaktadır. Bu doğruları güncellerken doğru dizi değerlerini f(net) dizi değerlerleri ile karşılaştırılmalıdır Burada eşit olmayan f(net)[i] ve d[i] değerleri için w[i] doğru dizi değeri güncellenmelidir. Noktalar yine Single Layer Continuous Learning'deki normalize edilmelidir. Bu kısım final ödevi ile birlikte teslim edilecek olup kodları final ödevi ile verilecektir.