Makine Öğrenmesi

Perceptron, backpropagation, aktivasyon ve PyTorch

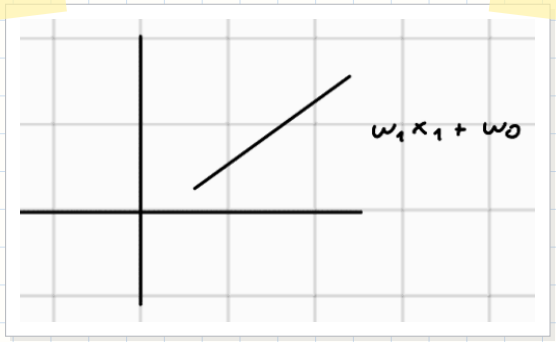
Orijinal not: Ders-8.pdf

UYARI Bu belge, el yazısı ders notlarından Claude (Anthropic) yardımıyla dijitale aktarılmıştır. Yardımcı kaynak olarak hazırlanmıştır; hata veya eksik içerebilir. Lütfen orijinal notları ve ders kaynaklarını esas alın.

NOTICE *This document was transcribed from handwritten lecture notes with the help of Claude (Anthropic). It is meant as a supplementary resource and may contain errors or omissions. Always check against the original notes and official course material.*

orj. s. 1

ROC curve ? okunamadı



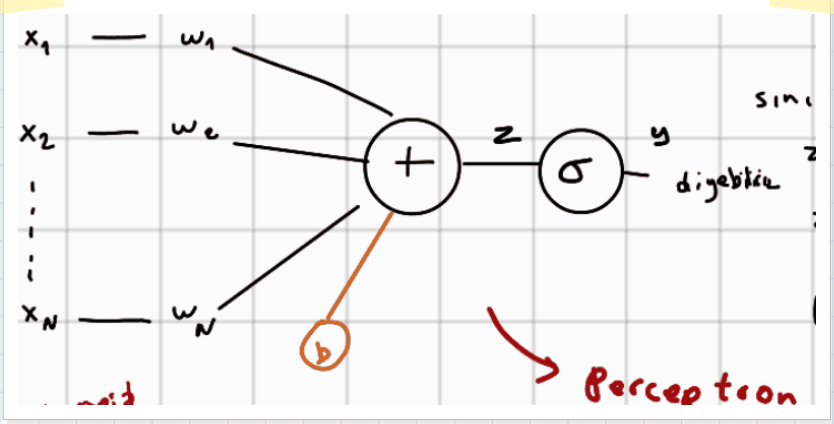
orijinal çizim, s. 1

$$\vec{w} \cdot \vec{x} + b = \hat{r}^t$$

$$\mathcal{L}(\theta|X) = \sum_{t=1}^{N} (\hat{r}^t - r^t)^2$$

türevini alıp 0'a ? eşitleriz

başka yöntem



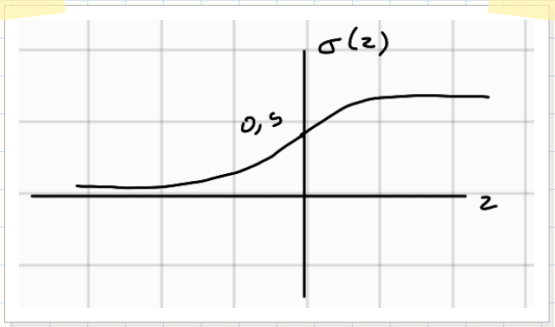
orijinal çizim, s. 1

sınıflandırmak için

$$z < 0 \Rightarrow C_0$$
$$z \geq 0 \Rightarrow C_1$$
$$P(C_0)$$

sigmoid

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



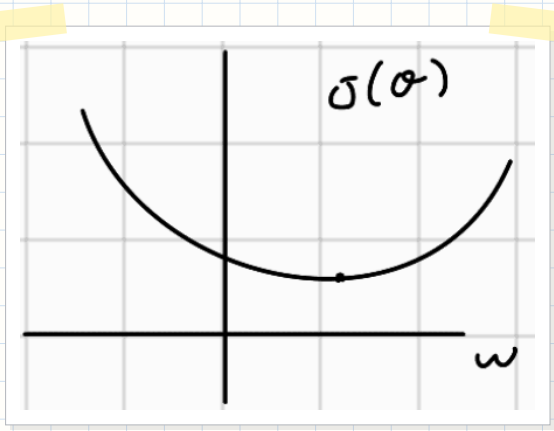
orijinal çizim, s. 1

$$z < 0 \Rightarrow C_0$$
$$z \geq 0 \Rightarrow C_1$$

Bu sefer iteratif olarak ilerliyoruz

gradient descent

$$w_i \leftarrow w_i - \eta \frac{\partial J}{\partial w_i} \quad \rightarrow \quad \text{maliyet fonksiyonu}$$



orijinal çizim, s. 1

orj. s. 2

Back Propagation

$$\hat{y}_i = \sigma(z_i) = \frac{1}{1+e^{-z_i}} \quad , \quad z_i = w^T \cdot x_i + b$$

$$\text{entropy} \quad \sum_{i=1}^{k} p_i \log p_i$$

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N} \underbrace{y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)}_{\text{cross entropy}}$$

$$\frac{\partial J(w)}{\partial w} = \frac{\partial J(w)}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial w}$$

? önce bu — sonra bu — sonra bu

$$\frac{\partial J(w)}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i} + \frac{1-y_i}{1-\hat{y}_i} \qquad \frac{\partial \hat{y}_i}{\partial z_i} = \hat{y}_i(1-\hat{y}_i)$$

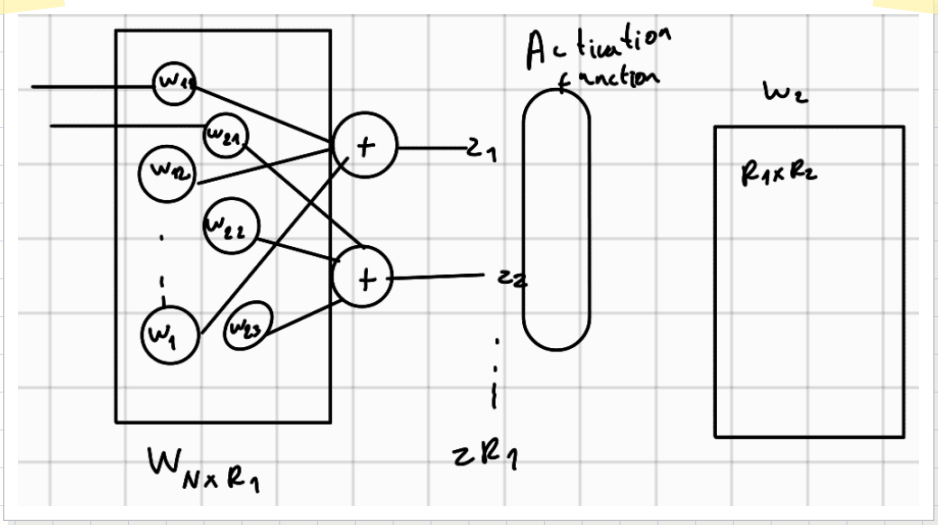
$$\frac{\partial z_i}{\partial w} = x_i$$

$$\frac{\partial J(w)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)x$$

sonra bunu ekleyeceğiz
geri sararak yapıyoruz

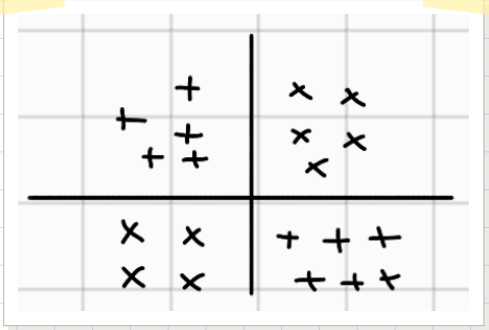
$$\frac{\partial J(w)}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)$$

$$X = \{x^t, r^t\}_{t=1}^{M} \qquad P(C_0) = \hat{y}_i \qquad P(C_1) = (1-\hat{y}_i)$$



orijinal çizim, s. 2

multi layer perceptron



orijinal çizim, s. 2

activation function

$$\phi(wx + b)$$

orj. s. 3

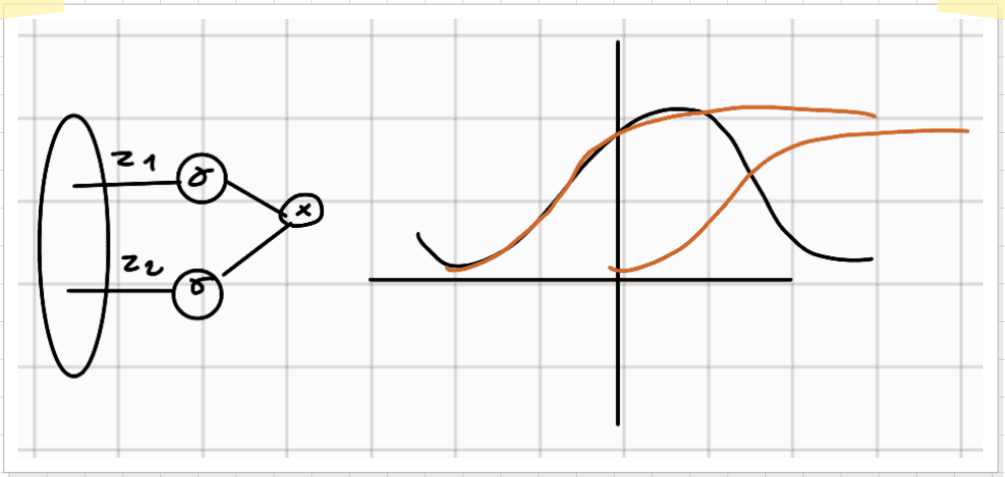
$$\Phi_2\left(W \sum \Phi_1(wx + b)\right)$$

Fourier'de

$$\langle \Phi_1(x), \Phi_2(x) \rangle = \int_{-\infty}^{\infty} \Phi_1(x)\, \Phi_2(x)\, dx = \overset{\text{ortogonal}}{0}$$

(Basis function)

ama burda



orijinal çizim, s. 3

Biz ortogonal olmalarını bekliyoruz

Böyle bir teorem var

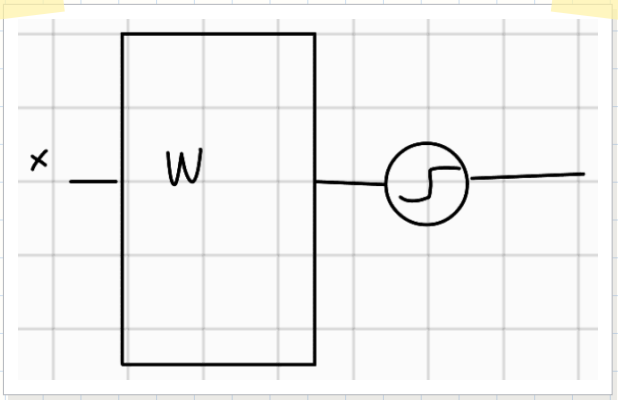
$$\sum_{i=1}^{\infty} w_i \, \Phi(x - x_i) \approx g(x)$$

yumuşak baz fonksiyonlarıyla mükemmel olarak tahmin edebiliriz

teorik olarak

universal approximator

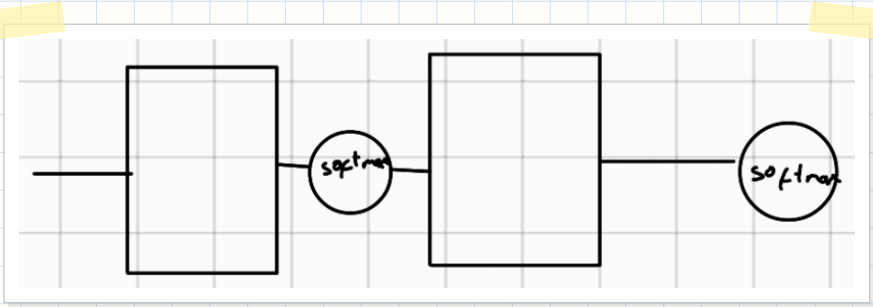
$$x \xrightarrow{f(x)} y$$



orijinal çizim, s. 3

① $\text{softmax} = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$ ↙ olasılık dağılımına ihtiyacımız olduğunda

orj. s. 4 ? soru



orijinal çizim, s. 4

böyle bir sıkıntı var sonsuza giderken türev 0 o yüzden back propagation iyi çalışmaz

bu yüzden genelde softmax en son katmanda kullanılır

② ReLU

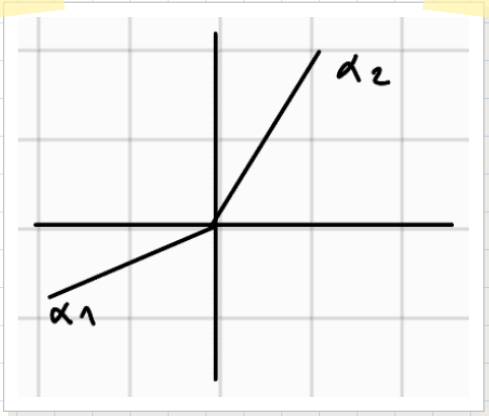


orijinal çizim, s. 4

ax + b şeklinde göstermiyoruz
non linear

negatif tarafta yine türevi 0, dead neuron sorununa yol açıyor

③ Leaky ReLU

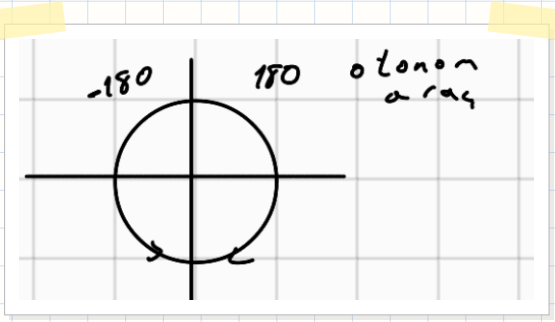


orijinal çizim, s. 4

nonlinearity daha az olduğu için daha fazla katman gerekiyor

$$180 \times [-1,1]$$

④ $\tanh = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



orijinal çizim, s. 4

bunda da türevler $+\infty, -\infty$'da 0'a yaklaşıyor

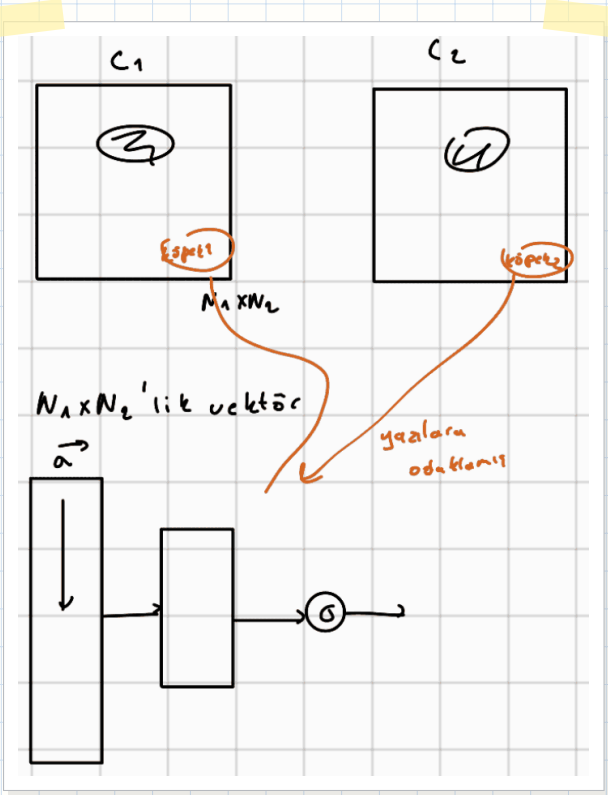
orj. s. 5

Derin öğrenmede katmanlar artar

① Tensorflow
② Pytorch

TensorDataset

Data augmentation



orijinal çizim, s. 5

Pseudo random

?x kütlesini ↙

DataScaler() → Standart Scaler()

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \qquad -1 < x_i < 1$$

$$10^6 - 10^{-4}$$
$$2 - 10^{-4}$$

orj. s. 6

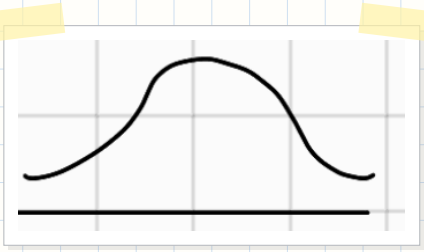
Scaler

① min-max

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

② Gauss $\mathcal{N}(0,1)$

$$\frac{x - \mu}{\sigma}$$



orijinal çizim, s. 6

sonra TensorDataset

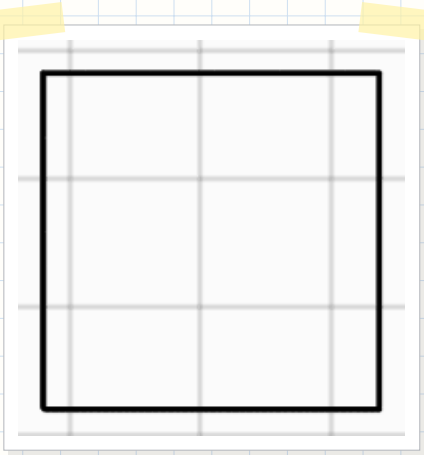
random split

Dataloader

class ANNmodel(nn.Module) ← bunun inherit ediyoruz

init : super().__init__()

pytorch linear
torch.nn.Linear(in_features, out_features, bias=True, device=None, dtype=None)



orijinal çizim, s. 6

$$\text{parametre sayısı} = \underset{\text{in}}{N} \times \underset{\text{out}}{M} + N$$

net = nn.Sequential

BatchNorm1d

Dropout

optim.Adam

nn.CrossEntropyLoss

orj. s. 7

```
def forward(self,x):
    x = x.to(self.device)
    x = x.to(T.float32)
```

out_ = F.one_hot(T.argmax(out,dim=1), num_classes=? k)

```
def learn:
self.optimizer.zero_grad()
  y_pred = self.forward(x)
    loss = self.loss(y_pred, y)
      loss.backward
```

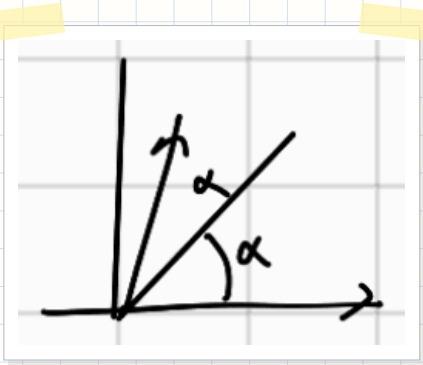
Normalization layer

BatchNorm1d

Dropout layers

overfit etmemek için bazı verileri düşürürüz

Cosine similarity



orijinal çizim, s. 7

$$\frac{x \cdot y}{\|x\| \, \|y\|} = \cos(\theta)$$

orj. s. 8

Loss Functions

L1 loss

MSE loss

KLDivLoss