Azərbaycan Texniki Universiteti

**Tələbə: Rəcəbov Sənan**

**Qrup: M665a4**

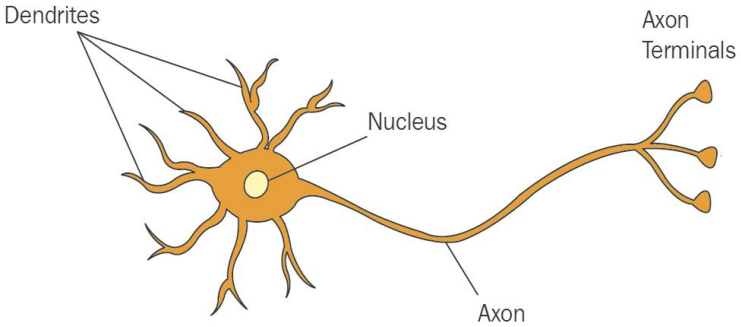
**Fənnin adı: Süni intellektin əsasları**

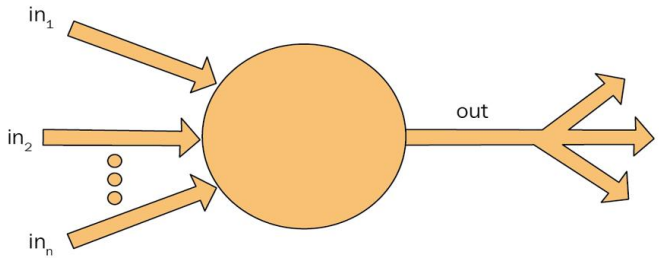
**Mövzu: Müxtəlif aktivləşmə funksiyalarının (sigmoid, ReLU, tanh) müqayisəli təhlili**

**Müəllim: Abbaslı Cavid**

# Süni neyron şəbəkələri

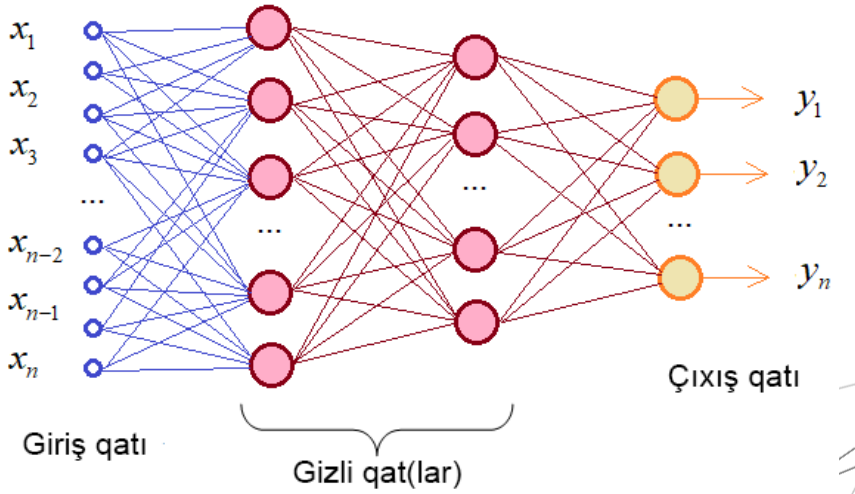
**Süni neyron şəbəkələri (Artificial Neural Networks – ANN)** insan beynindəki neyronlar və sinapslardan ilham alaraq yaradılmış riyazi modellərdir. Bu şəbəkələr məlumatları qəbul edir, emal edir və nəticə çıxarır. Onlar öyrənə, adaptasiya ola və əvvəlki nümunələrə əsaslanaraq yeni məlumatlara cavab verə bilirlər.



# Süni neyron şəbəkələri

Neyronlar bir-biri ilə əlaqəli olduqlarına görə, bir neyronun nəticəsi əlaqəli neyrona giriş məlumatı kimi ötürülə bilər. Beləliklə, şəbəkə formalaşır.



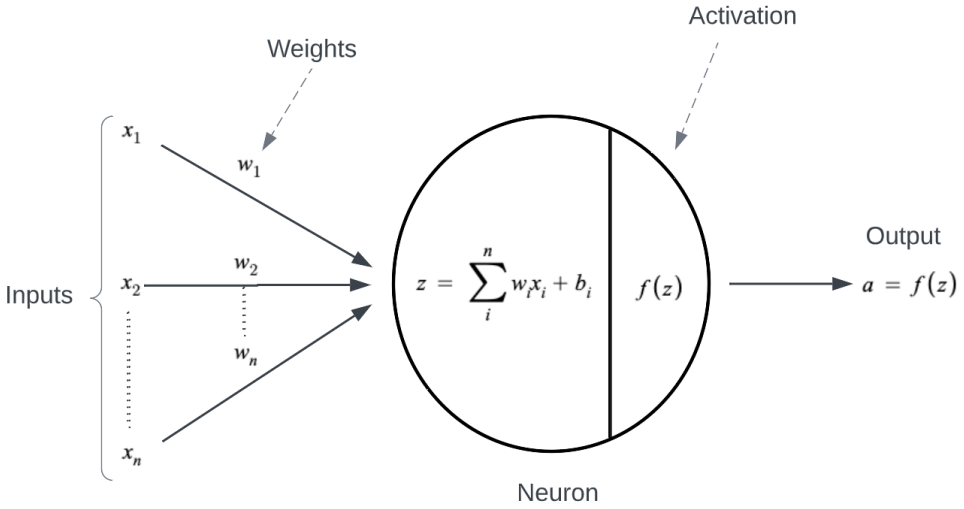

ANN-lər 3 əsas qatdan ibarətdir:

- Giriş qatı: X xarakterli dəyişənləri qəbul edir.
- Gizli qat(lar): Hesablama aparan əsas hissədir.
- Çıxış qatı: Proqnoz və ya qərar burada verilir.

Şəbəkədə qatların sayı və neyronların miqdarı modelin **öyrənmə gücünü** və **ümumiləşdirmə bacarığını** müəyyən edir.

# Süni neyron şəbəkələri

Hər bir süni neyron bir neçə giriş qəbul edir ($x_1, x_2, \ldots, x_n$), bu girişlərə uyğun çəkilər ($w_1, w_2, \ldots, w_n$) təyin edir, bunları toplayaraq üzərinə bir bias ($b$) əlavə edir. Bu nəticəyə aktivasiya funksiyası tətbiq olunaraq çıxış ($y$) alınır.




ANN riyazi interpretasiyası:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b \implies \hat{y} = f(z)$$

## Süni neyron şəbəkələri

**Aktivasiya funksiyası** neyron qatında xətti çevrilmədən sonra tətbiq olunan və neyronun çıxışını formalaşdıran **qeyri-xətti funksiyadır**. Aktivasiya funksiyası modelə qeyri-xəttilik gətirir və mürəkkəb nümunələri öyrənməyə imkan verir. Praktikada ən çox istifadə olunan aktivasiya funksiyaarı aşağıdakılardır:

- **Sigmoid:** 0 ilə 1 arasında çıxış verir, ehtimallar üçün uyğundur.

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

- **ReLU:** Mənfi girişləri sıfıra çevirir, sürətli öyrənmə təmin edir

$$f(z) = max(0, z)$$

- **tanh:** -1 ilə +1 arasında çıxış verir, mərkəzləşdirilmiş siqnallarla daha balanslıdır.

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

## Süni neyron şəbəkələri

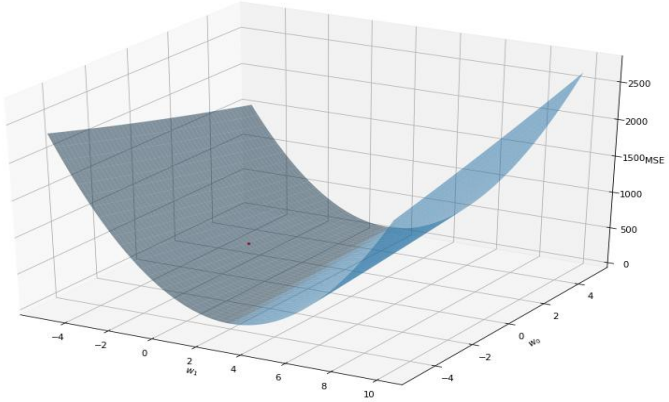
Aktivasiya funksiyalarının müqayisəsinə keçməzdən əvvəl iki vacib anlayışı nəzərdən keçirmək istəyirəm:

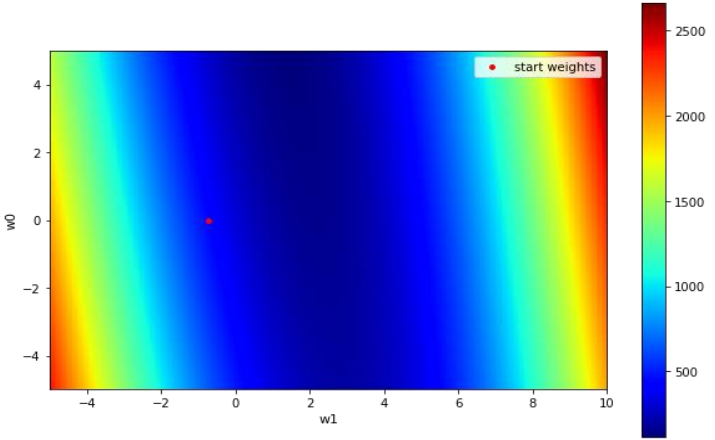
- **Qradient enişi (Gradient Descent)** - itki funksiyasını *J(w)* minimuma endirmək üçün parametrleri *w* istiqamətli törəmənin (qradiyentin) əksi üzrə addım-addım yeniləyən optimallaşdırma üsuludur.

- **Xətanın geri yayılması (Backpropagation) alqoritmi** - şəbəkənin öyrənmə üsuludur.

# Gradient Descent

**Qradient enişi** — funksiyanın lokal **minimumunu** və ya **maksimumunu** qradient boyunca hərəkət etməklə tapma üsuludur.




*f* funksiyasının qradienti onun qismən törəmələrindən ibarət *n*-ölçülü vektordur.

$$\nabla f(x, y, z) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$$

## Gradient Descent

**Baş düstur (Update qaydası)** — prosesin mərkəzi hissəsidir: daha yaxşı nəticə üçün köhnə parametrlərdən yenilərinə necə keçməyi izah edir.

$$w_{new} = w_{old} - \eta \cdot \nabla J(w_{old})$$

Qradient funksiyanın ən sürətli artım istiqamətini göstərir. Deməli, antiqradiyent onun ən sürətli azalma istiqamətini göstərəcək. Buna görə ∇*J* mənfi götürürük.

Burada $\eta$öyrənmə sürətidir – daha sürətli və ya daha yavaş öyrənmə üçün tənzimlənir.

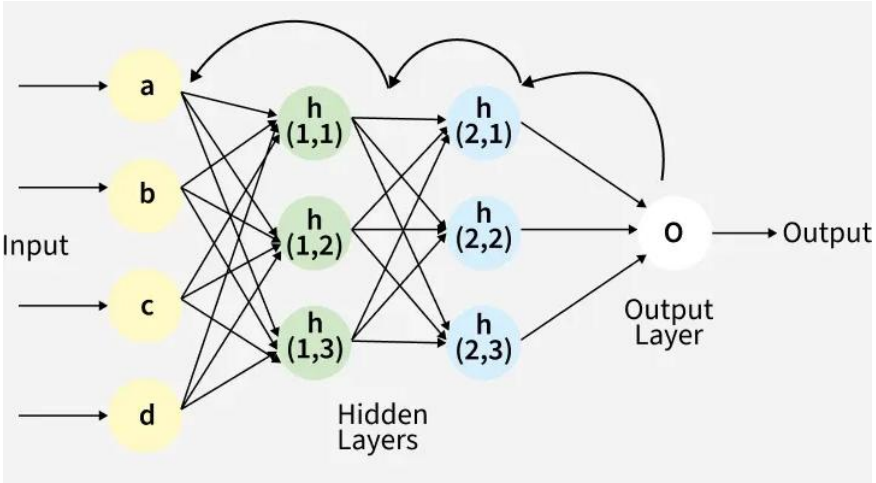
Qradienti (∇*J*) tapmaq üçün — yəni istənilən çəki *w* dəyişəndə *J*-nin necə dəyişdiyini müəyyən etmək üçün — qatların bütün zənciri üzrə geriyə “keçməliyik”.

# Xətanın geri yayılması (Backpropagation) alqoritmi

**Backpropagation alqoritmi** – neyron şəbəkələrdə parametrlərin (çəkilər və bias) **gradient descent** üsulu ilə optimallaşdırılması üçün istifadə olunur. Bu metod şəbəkənin proqnozları ilə real nəticələr arasındakı fərqi (xətanı) azaldır və nəticədə model daha dəqiq olur.

**Əsas ideya**

1. **Forward pass (irəli ötürülmə):**
   girişlər qat-qat ötürülürvə çıxış hesablanır.
2. **Backward pass (geri yayılma):**
   çıxışdakı xəta geri istiqamətdə yayılır və çəkilər yenilənir.



# Xətanın geri yayılması (Backpropagation) alqoritmi

**Backpropagation alqoritminin addımları**

**1. İrəli keçid:** Girişlərdən çıxışa doğru hesablamalar aparılır.

**2.Xəta hesablanması:** itki funksiyası ilə çıxış və real dəyər arasındakı fərq tapılır.

$$Loss = \frac{1}{2}(y - \hat{y})^2$$

**3.Geri yayılma:** Zəncir qaydası ilə error çıxışdan geriyə ötürülür.

**4.Çəkilərin yenilənməsi:** gradient descent ilə çəkilər və biaslar dəyişdirilir.

$$w_{i_{new}} = w_i - \eta \cdot \frac{\partial Loss}{\partial w_i}$$

**Məqsədimiz:** $w_i$-ni elə dəyişməkdir ki, **minimum** olsun.

# Xətanın geri yayılması (Backpropagation) alqoritmi

**Chain rule**

Çox qatlı şəbəkələrdə bir qatın çıxışı digər qatın girişi olduğuna görə çəkilərin törəməsini hesablamaq üçün **mürəkkəb funksiyanın xüsusi törəməsi** qaydasından istifadə olunur.

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_i}$$

Xüsusi törəmə: $\hat{y} \to z \to w_i$

# Sigmoid funksiyası

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

Törəmə: $\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$

Çıxış aralığı: (0, 1)

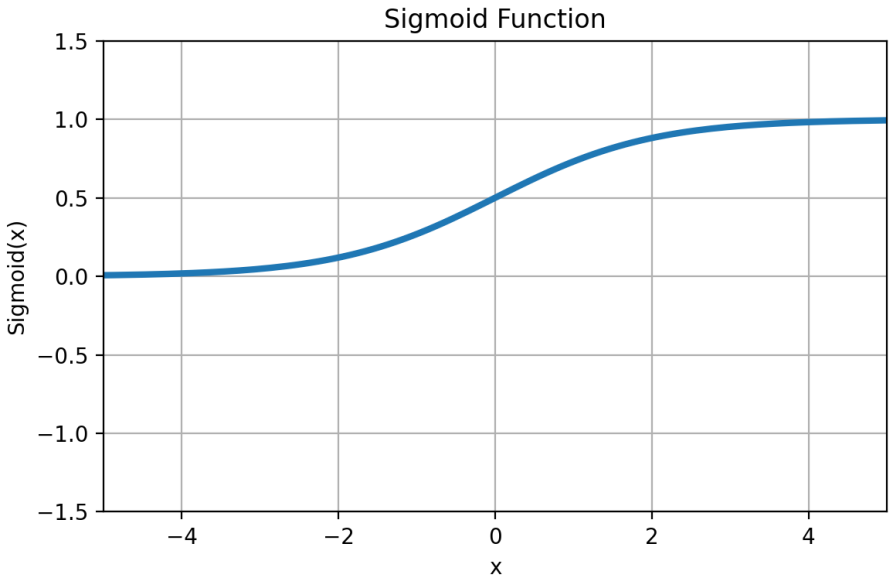
Üstünlükləri:

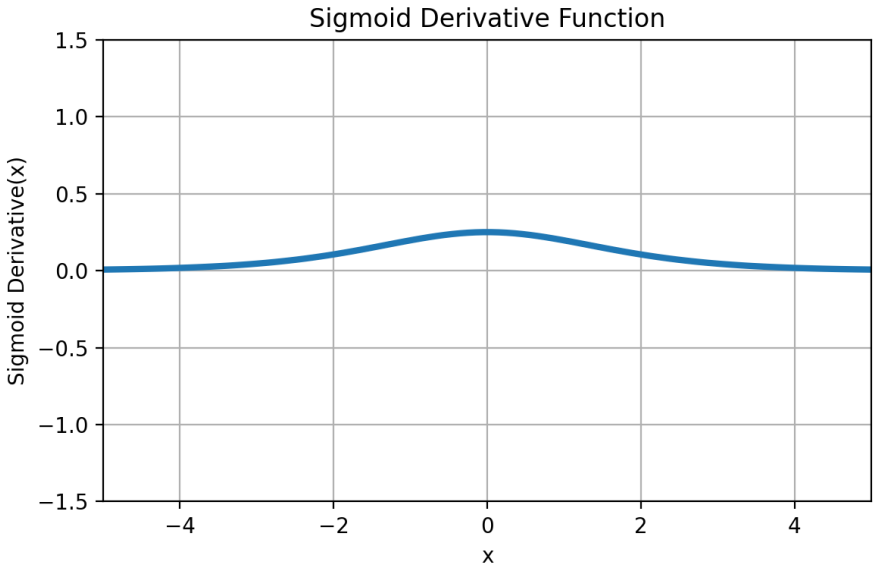
- Çıxış dəyərləri həmişə müsbət və məhduddur — ehtimal kimi götürələ bilər.
- Kiçik neyron şəbəkələrində və çıxış qatında (məsələn, binary classification) uyğun işləyir.

Mənfi cəhətləri:

- Gradientin itməsi problemi (vanishing gradient): Giriş çox böyük və ya kiçik olduqda törəmə sıfıra yaxınlaşır — bu da şəbəkənin öyrənməsini yavaşladır.
- Çıxış dəyərləri 0-a mərkəzli deyil → təlim qeyri-sabit ola bilər

# Sigmoid funksiyası



# ReLU (Rectified Linear Unit) funksiyası

ReLU funksiyası: $f(z) = max(0, z)$

Törəmə: $f'(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$

Çıxış aralığı: [0, +∞)

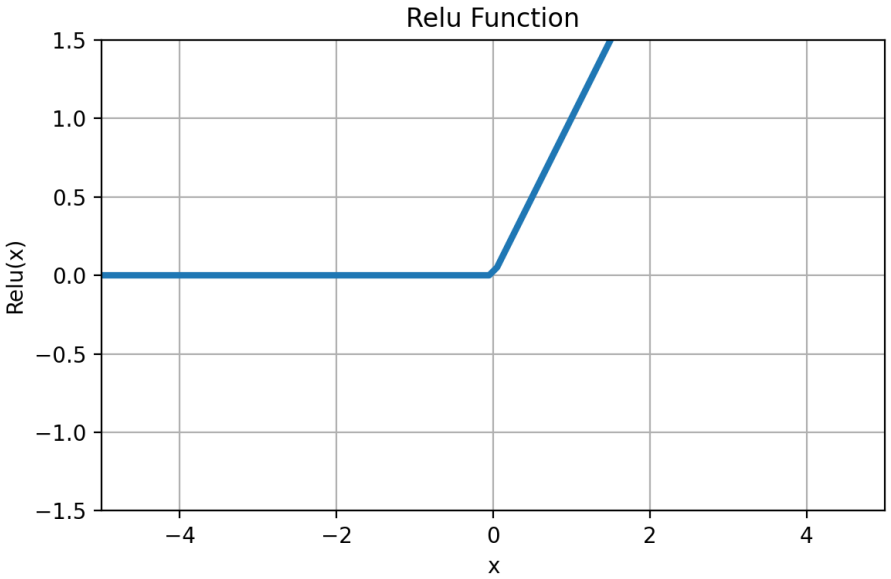
Üstünlükləri:

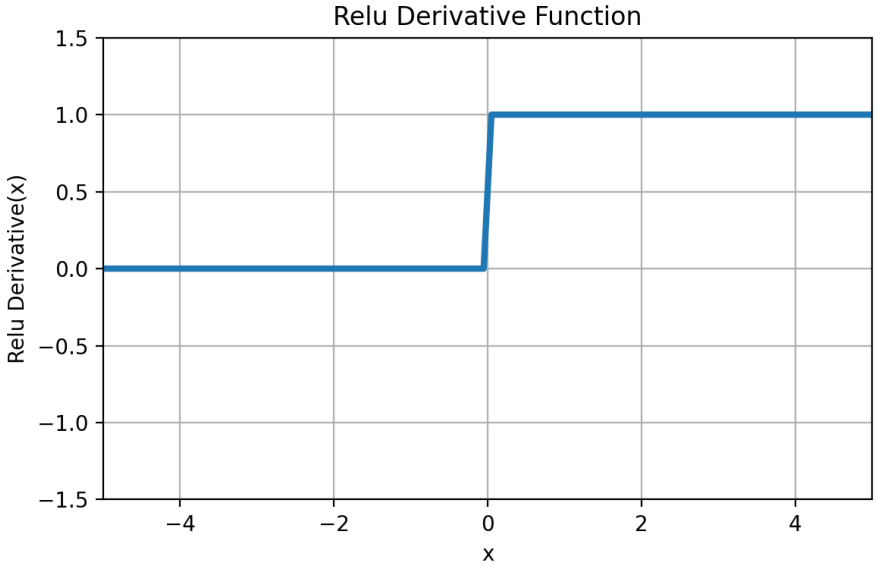
- Hesablama baxımından çox sadə və sürətlidir.
- Dərin şəbəkələrdə öyrənməni mümkün edir (gradient ötürülməsi saxlanılır).

Mənfi cəhətləri:

- Negativ girişlərdə törəmə sıfır olur: bu, “dead neuron” probleminə səbəb ola bilər.
- Çıxışlar sıfır mərkəzli deyil – bu, bəzi hallarda öyrənmədə asimmetriya yarada bilər.

# ReLU (Rectified Linear Unit) funksiyası



# Tanh (Hiperbolik tangens) funksiyası

ReLU funksiyası: $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Törəmə: $f'(z) = 1 - f(z)^2$

Çıxış aralığı: (-1, 1)

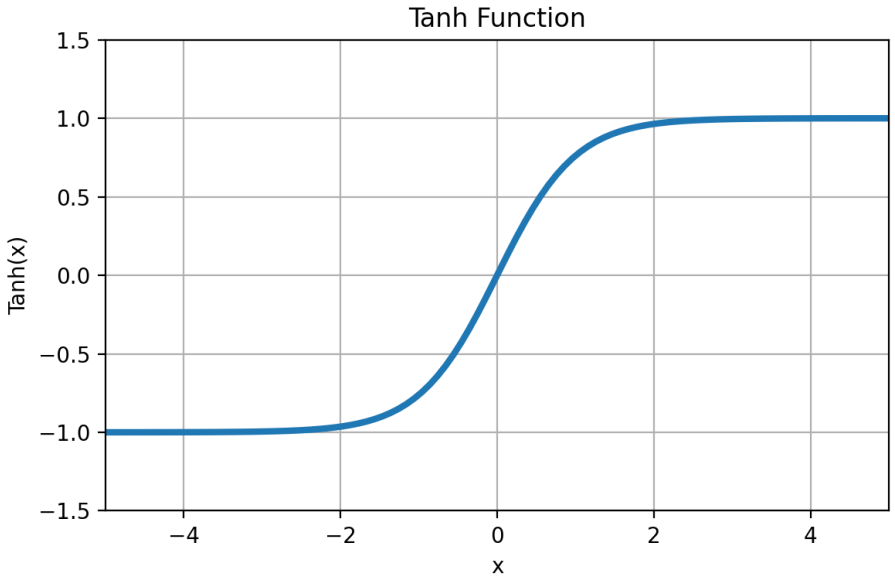
Üstünlükləri:

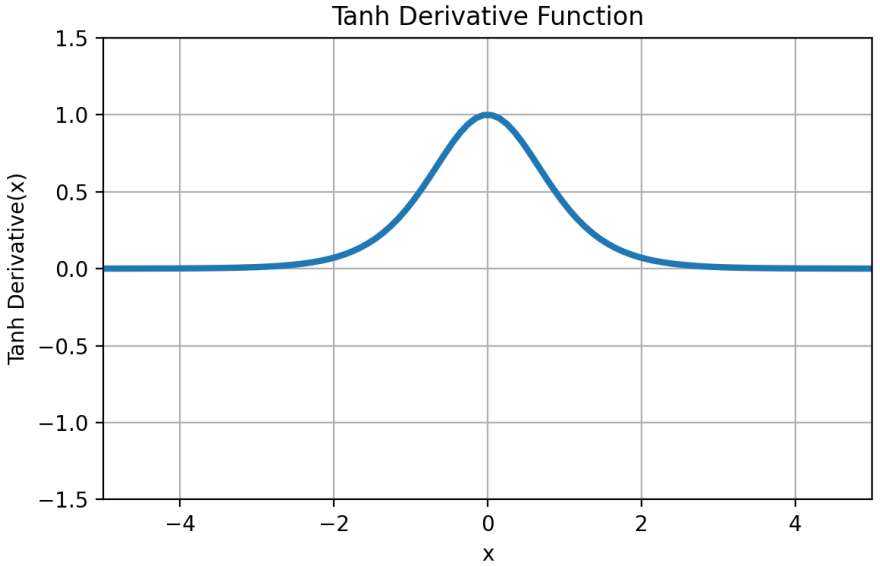
- Aktivasiyaların ortalaması sıfıra yaxın olur — bu, daha balanslı və effektiv öyrənməyə səbəb olur.

Mənfi cəhətləri:

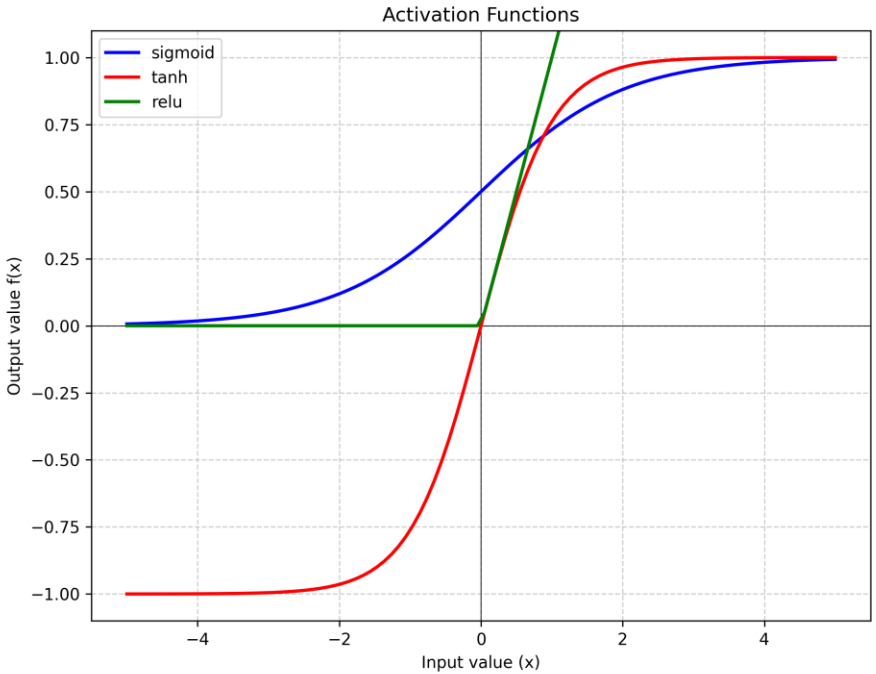
- Dərin şəbəkələrdə gradientin zəifləməsi problemi qalır.
- Sigmoid kimi doymuş sahələrdə neyronun aktivliyi azalır və bu öyrənməni zəiflədir.

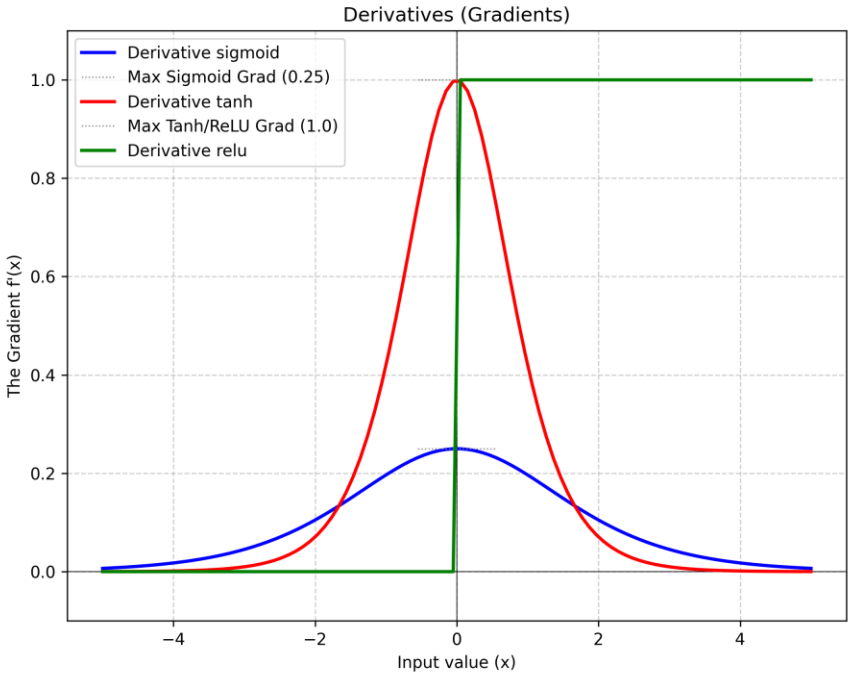
# Tanh (Hiperbolik tangens) funksiyası



# Sigmoid, Relu & Tanh



# Aktivləşmə funksiyaları - Python code

Aktivasiya funksiyalarının Python-da NumPy vasitəsilə reallaşdırılması.

```python
def relu(x: NDArray) -> NDArray:
    return np.maximum(x, 0)


def sigmoid(x: NDArray) -> NDArray:
    return 1 / (1 + np.exp(-x))


def tanh(x: NDArray) -> NDArray:
    return np.tanh(x)
```

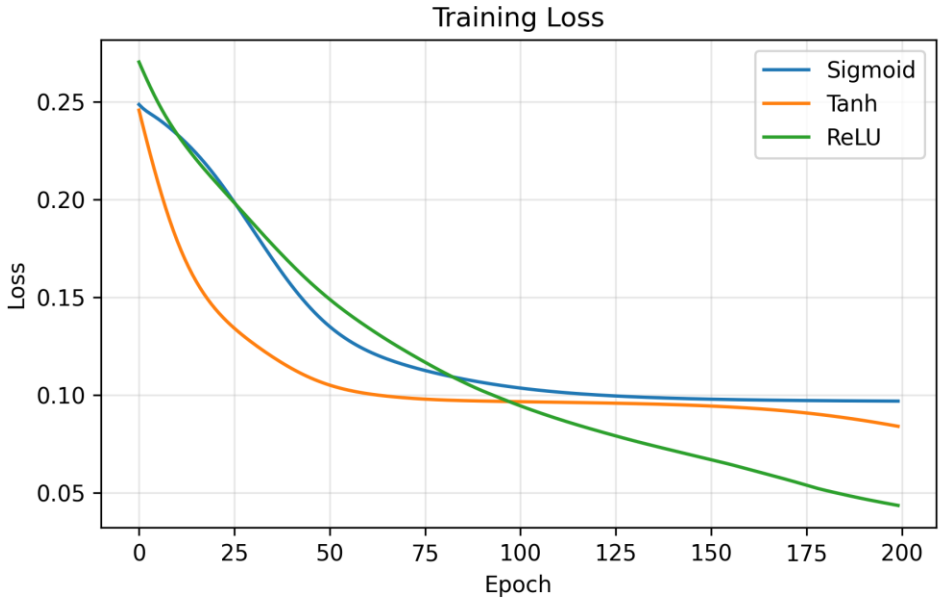
```python
def sigmoid_derivative(x: NDArray) -> NDArray:
    s = sigmoid(x)
    return s * (1 - s)


def tanh_derivative(x: NDArray) -> NDArray:
    return 1.0 - np.tanh(x) ** 2


def relu_derivative(x: NDArray) -> NDArray:
    return (x > 0).astype(int)

```

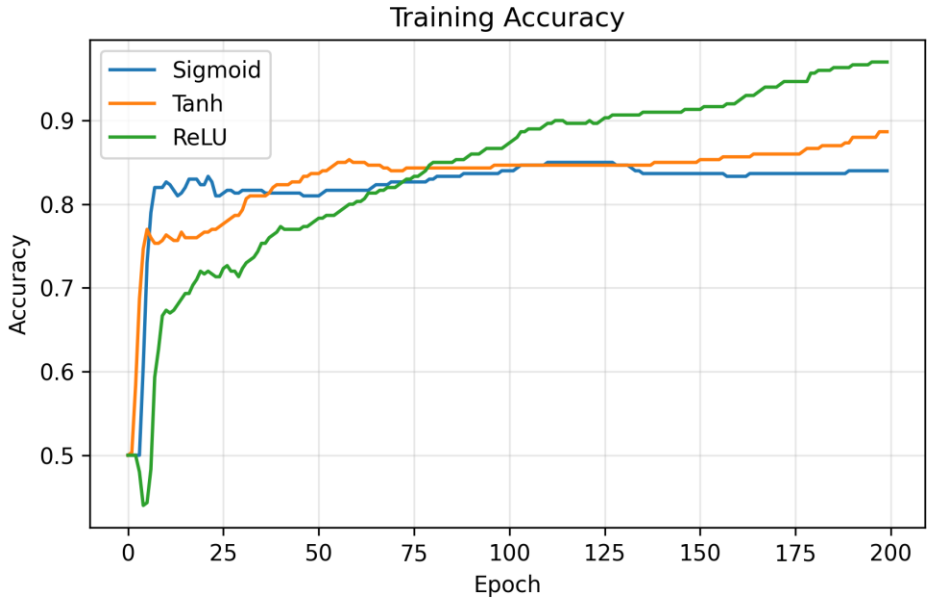
# Cədvəl şəklində ümumi müqayisəsi

| Xüsusiyyət | Sigmoid | Tanh | ReLU |
|---|---|---|---|
| Çıxış aralığı | (0; 1) | (-1; 1) | [0; ∞) |
| Sıfır mərkəzli? | Xeyr | Bəli | Xeyr |
| Maksimum törəmə | 0.25 | 1 | 1 (yalnız (x > 0)) |
| Saturasiya (doyma)? | Bəli | Bəli | Yalnız (x < 0) |
| Gradientin zəifləməsi | Güclü | Orta | Yox (pozitiv sahədə) |
| Dead neuron problemi | Yox | Yox | Bəli |
| Hesablama mürəkkəbliyi | Orta (exp) | Orta (exp) | Aşağı (maksimum) |
| Tətbiq sahələri | Çıxış, LSTM qapıları | RNN, LSTM daxili qat | Dərin gizli qatlar |

## Müqayisəsi - Python

Bu sadə PyTorch modeli Sigmoid, Tanh və ReLU aktivasiya funksiyalarının öyrənmə nəticələrini müqayisə edir.




Code: https://github.com/sananradjabov/ANN-ActivationFunction-sigmoid-relu-tanh-AzTU

# Diqqətinizə görə təşəkkürlər!