EOC
EUROASIAN
ONLINE
CONFERENCES

GERMANY
CONFERENCE

INTERNATIONAL CONFERENCE ON
SCIENCE, ENGINEERING AND TECHNOLOGY

Google Scholar zenodo OpenAIRE doi digital object identifier

INTERNATIONAL CONFERENCE ON SCIENCE, ENGINEERING AND TECHNOLOGY: a collection scientific works of the International scientific conference – Gamburg, Germany, 2025 Issue 6

Languages of publication: Uzbek**,** English, Russian, German, Italian, Spanish,

The collection consists of scientific research of scientists, graduate students and students who took part in the International Scientific online conference **« INTERNATIONAL CONFERENCE ON SCIENCE, ENGINEERING AND TECHNOLOGY ».** Which took place in Gamburg, 2025.

Conference proceedings are recommended for scientists and teachers in higher education establishments. They can be used in education, including the process of post - graduate teaching, preparation for obtain bachelors' and masters' degrees. The review of all articles was accomplished by experts, materials are according to authors copyright. The authors are responsible for content, researches results and errors.

PERSEPTRONNI O‘RGATISH ALGORITMI

Tojimamatov Isroil Nurmamatovich
Farg‘ona davlat universiteti Amaliy matematika
va informatika kafedrasi katta o’qituvchisi
E-mail: istailtojimamatov@gmail.com
Muhammadshokirova Dinora Ma'rufjon qizi
Farg‘ona davlat universiteti Amaliy matematika
yo‘nalishi 3-bosqich 23.07-guruh talabasi
E-mail: dinorashermurodova91@gmail.com

Annotatsiya. Ushbu tezisda sun'iy intellektning asosiy modellaridan biri — perseptronning o‘rgatish jarayoni, algoritmlari va matematik asoslari batafsil tahlil qilinadi. Rosenblattning dastlabki o‘rgatish qoidasidan tortib, gradient tushish va stoxastik gradient tushish usullarigacha bo‘lgan o‘rgatish metodlari ko‘rib chiqilgan. Shuningdek, turli aktivatsiya funktsiyalari yordamida perseptronni samarali o‘rgatish usullari ham ko‘rsatildi. Python dasturlash tilida yozilgan amaliy kodlar orqali modelning ishlash printsipi va o‘rgatish jarayoni namoyish etilgan.

Kalit so‘zlar: perseptron, sun'iy neyron tarmoq, o‘rgatish algoritmi, gradient tushish, stoxastik gradient tushish, aktivatsiya funktsiyasi, Python dasturlash, mashina o‘rganish, log-loss, sigmoid, ReLU.

Abstract. This thesis presents a detailed analysis of one of the primary models of artificial intelligence, specifically the learning process, algorithms, and mathematical foundations of the perceptron. It examines teaching methods ranging from Rosenblatt's initial teaching rule to gradient drop and stochastic gradient drop methods. Methods for effective perceptron training using various activation functions were also shown. The operating principle of the model and the learning process are demonstrated using application codes written in the Python programming language.

Keywords: perceptron, artificial neural network, learning algorithm, gradient drop, stochastic gradient drop, activation function, Python programming, machine learning, log-loss, sigmoid, ReLU.

Аннотация. В данном тезисе подробно анализируются процесс обучения, алгоритмы и математические основы одной из основных моделей искусственного интеллекта - перцептрона. Рассматриваются методы обучения, начиная с исходного правила обучения Розенблатта и заканчивая методами градиентного спуска и стохастического градиентного спуска. Также показаны методы эффективного обучения перцептрона с помощью различных функций активации. С помощью прикладных кодов, написанных на языке программирования Python, демонстрируется принцип работы модели и процесс обучения.

Ключевые слова: perceptron, искусственная нейронная сеть, алгоритм обучения, градиентный спуск, стохастический градиентный спуск,

функция активации, программирование Python, машинное обучение, log-loss, сигмоид, ReLU.

Kirish

Sun’iy intellekt (SI) sohasida neyron tarmoqlar va ularning asosiy elementlari bo‘lgan perseptron modeli muhim o‘rin tutadi. Perseptron — bu oddiy sun’iy neyron bo‘lib, kirish ma’lumotlarini qabul qilib, ularni vaznlar bilan ko‘paytiradi, keyin yig‘indini hisoblab, aktivatsiya funktsiyasi orqali chiqish signalini hosil qiladi. Ushbu model mashina o‘rganish jarayonining asosiy bloklaridan biri sifatida, tasniflash va bashorat qilish vazifalarini bajarishda keng qo‘llaniladi. Perseptronni o‘rgatish jarayoni esa uning vazn va bias qiymatlarini ma’lumotlar asosida moslashtirishdan iborat bo‘lib, natijada model yangi, ko‘rilmagan ma’lumotlarga nisbatan ham to‘g‘ri javob bera oladi. O‘rgatish algoritmlari, xususan, Rosenblattning klassik qoidasidan tortib, zamonaviy gradient tushish va stoxastik gradient tushish usullarigacha bo‘lgan texnikalar mavjud. Bundan tashqari, turli aktivatsiya funktsiyalari — pog‘onali, sigmoid, tanh va ReLU — modelning moslashuvchanligi va aniqligini oshirishga yordam beradi. Ushbu tezisda perseptronning o‘rgatish algoritmlari, matematik asoslari hamda Python dasturlash tilida amaliy qo‘llanishi batafsil ko‘rib chiqiladi. Bu esa yangi boshlovchilar va mutaxassislar uchun sun’iy neyron tarmoqlarini chuqurroq tushunish va amaliy loyihalarda qo‘llash imkonini yaratadi.

Perseptronni o'rgatish jarayoni asosan modelning vazn va bias qiymatlarini yangilash orqali bashoratdagi xatolikni kamaytirishga qaratilgan. Eng oddiy perseptron o'rgatish qoidasi Rosenblatt tomonidan taklif qilingan bo‘lib, u quyidagi asosiy bosqichlarni o‘z ichiga oladi:

- Kirish ma'lumotlari asosida chiqishni bashorat qilish
- Bashorat qilingan natija va haqiqiy natija o‘rtasidagi xatolikni hisoblash
- Vazn va bias qiymatlarini xatolik asosida yangilash
- Xatolik yo‘qolgunga qadar jarayonni takrorlash

Quyidagi kodda Rosenblattning asosiy o‘rgatish jarayoni ko‘rsatilgan:

```
import numpy as np
class Perceptron:
  def __init__(self, input_size, learning_rate=0.01, max_epochs=1000):
    self.weights = np.zeros(input_size)
    self.bias = 0
    self.learning_rate = learning_rate
    self.max_epochs = max_epochs
    self.errors_history = []
      def predict_sample(self, x):
    activation = np.dot(x, self.weights) + self.bias
    return 1 if activation >= 0 else 0
```

```
def predict(self, X):
    return np.array([self.predict_sample(x) for x in X])

def train (self, X, y):
    n_samples = X.shape[0]
    for epoch in range(self.max_epochs):
        errors = 0
        for i in range(n_samples):
            x_i = X[i]
            y_i = y[i]
            y_pred = self.predict_sample(x_i)
            error = y_i - y_pred
            if error != 0:
                self.weights += self.learning_rate * error * x_i
                self.bias += self.learning_rate * error
                errors += 1
        self.errors_history.append(errors)
        if errors == 0:
            print(f"O'rgatish tugallandi: {epoch+1} ta iteratsiyadan so'ng.")
            break
    return self
```

Ushbu usulda har bir ma'lumot namunasi uchun bashorat qilinib, agar natija xato bo‘lsa, vazn va bias qiymatlari yangilanadi. Jarayon barcha namunalar to‘g‘ri tasniflangan holatga yetguncha yoki maksimal iteratsiya sonigacha davom etadi.

Xatolik funktsiyasi va gradient tushish

Rosenblatt qoidasidan tashqari, zamonaviy perseptron o‘rgatish algoritmlari ko‘pincha gradient tushish (gradient descent) usulini qo‘llaydi. Bu usul xatolik funktsiyasini (masalan, log-loss) minimal qilish orqali model parametrlarini optimallashtiradi. Quyidagi kodda gradient tushish asosida o‘rgatiladigan sigmoid aktivatsiyaga ega perseptron misoli keltirilgan:

```
class GradientPerceptron:
    def __init__(self, input_size, learning_rate=0.01, max_epochs=1000):
        self.weights = np.zeros(input_size)
        self.bias = 0
        self.learning_rate = learning_rate
        self.max_epochs = max_epochs
        self.loss_history = []

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))
```

```
  def predict_proba(self, X):
    return self.sigmoid(np.dot(X, self.weights) + self.bias)

  def predict(self, X):
    return (self.predict_proba(X) >= 0.5).astype(int)

  def compute_loss(self, X, y):
    y_pred = self.predict_proba(X)
    return -np.mean(y * np.log(y_pred + 1e-10) + (1 - y) * np.log(1 - y_pred + 1e-10))

  def train(self, X, y):
    n_samples = X.shape[0]
    for epoch in range(self.max_epochs):
      y_pred = self.predict_proba(X)
      loss = self.compute_loss(X, y)
      self.loss_history.append(loss)
      dw = (1/n_samples) * np.dot(X.T, (y_pred - y))
      db = (1/n_samples) * np.sum(y_pred - y)
      self.weights -= self.learning_rate * dw
      self.bias -= self.learning_rate * db
      if epoch % 100 == 0:
        print(f"Epoch {epoch}, Loss: {loss:.6f}")
      if loss < 1e-5:
        print(f"Konvergensiya: {epoch+1} ta iteratsiyadan so'ng.")
        break
    return self
```

Gradient tushish algoritmi orqali parametrlar xatolik gradientiga qarshi yo'nalishda yangilanadi va shu bilan model yaxshilanadi.

Turli aktivatsiya funktsiyalari bilan perseptronni o'rgatish

Perseptronlarda turli aktivatsiya funktsiyalari qo‘llanilishi mumkin. Eng oddiy — pog'onali (step) funktsiya bo‘lsa, zamonaviy modellarda sigmoid, tanh, yoki ReLU kabi noaniq va qiyshiq funktsiyalar ishlatiladi. Bu aktivatsiya funktsiyalarining hosilalari gradient tushish jarayonida muhim rol o‘ynaydi. Quyida turli aktivatsiya funktsiyalar va ularning hosilalari bilan ishlaydigan moslashuvchan perseptron klassining namunasi keltirilgan:

```
FlexiblePerceptron:
  def __init__(self, input_size, activation_function='sigmoid', learning_rate=0.01, max_epochs=1000):
    self.weights = np.zeros(input_size)
    self.bias = 0
    self.learning_rate = learning_rate
```

```
        self.max_epochs = max_epochs
        self.loss_history = []

        if activation_function == 'step':
            self.activation = lambda x: np.where(x >= 0, 1, 0)
            self.activation_derivative = lambda x: 0
        elif activation_function == 'sigmoid':
            self.activation = lambda x: 1 / (1 + np.exp(-x))
            self.activation_derivative = lambda x: x * (1 - x)
        elif activation_function == 'tanh':
            self.activation = lambda x: np.tanh(x)
            self.activation_derivative = lambda x: 1 - x**2
        elif activation_function == 'relu':
            self.activation = lambda x: np.maximum(0, x)
            self.activation_derivative = lambda x: np.where(x > 0, 1, 0)
        else:
            raise ValueError("Noto'g'ri aktivatsiya funktsiyasi")

    def forward(self, X):
        self.z = np.dot(X, self.weights) + self.bias
        self.a = self.activation(self.z)
        return self.a

    def compute_loss(self, y_true, y_pred):
        return np.mean((y_true - y_pred) ** 2)

    def train(self, X, y):
        n_samples = X.shape[0]
        for epoch in range(self.max_epochs):
            y_pred = self.forward(X)
            loss = self.compute_loss(y, y_pred)
            self.loss_history.append(loss)
            delta = (y_pred - y) * self.activation_derivative(y_pred)
            dw = (1/n_samples) * np.dot(X.T, delta)
            db = (1/n_samples) * np.sum(delta)
            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db
            if epoch % 100 == 0:
                print(f"Epoch {epoch}, Loss: {loss:.6f}")
            if loss < 1e-5:
                print(f"Konvergensiya: {epoch+1} ta iteratsiyadan so'ng.")
                break
```

```
        return self
    def predict(self, X):
        return (self.forward(X) >= 0.5).astype(int)
```

Perseptronni o'rgatish metodlari va amaliy jihatlar

Stoxastik gradient tushish (SGD)

Stoxastik gradient tushish algoritmi (SGD) har bir iteratsiyada faqat bitta ma'lumot namunasi asosida parametrlarni yangilaydi. Bu usul o‘rgatishni tezlashtiradi va katta hajmdagi ma'lumotlar bilan ishlashda samarali hisoblanadi. Quyida SGD asosidagi perseptron o'rgatish jarayoni misoli keltirilgan:

```
class SGDPerceptron:
    def __init__(self, input_size, learning_rate=0.01, max_epochs=1000):
        self.weights = np.zeros(input_size)
        self.bias = 0
        self.learning_rate = learning_rate
        self.max_epochs = max_epochs
        self.loss_history = []

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def forward(self, x):
        z = np.dot(x, self.weights) + self.bias
        return self.sigmoid(z)

    def compute_sample_loss(self, y_true, y_pred):
        return -y_true * np.log(y_pred + 1e-10) - (1 - y_true) * np.log(1 - y_pred +
1e-10)
    def train(self, X, y):
        n_samples = X.shape[0]
        indices = np.arange(n_samples)
        for epoch in range(self.max_epochs):
            total_loss = 0
            np.random.shuffle(indices)
            X_shuffled = X[indices]
            y_shuffled = y[indices]
            for i in range(n_samples):
                x_i = X_shuffled[i]
                y_i = y_shuffled[i]
                y_pred = self.forward(x_i)
                loss = self.compute_sample_loss(y_i, y_pred)
```

```
            total_loss += loss
            gradient = y_pred - y_i
            self.weights -= self.learning_rate * gradient * x_i
            self.bias -= self.learning_rate * gradient
        avg_loss = total_loss / n_samples
        self.loss_history.append(avg_loss)
        if epoch % 100 == 0:
            print(f"Epoch {epoch}, Loss: {avg_loss:.6f}")
        if avg_loss < 1e-5:
            print(f"Konvergensiya: {epoch+1} ta iteratsiyadan so'ng.")
            break
    return self
def predict(self, X):
    return np.array([1 if self.forward(x) >= 0.5 else 0 for x in X])
```

Xulosa

Perseptron — sun'iy neyronlarning eng sodda va asosiy modeli bo‘lib, uning o‘rgatish jarayoni mashina o‘rganish va sun'iy intellekt sohasida katta ahamiyatga ega. Rosenblatt algoritmi asosida boshlanib, keyinchalik gradient tushish va stoxastik gradient tushish kabi optimallashtirish usullari rivojlandi. Turli aktivatsiya funktsiyalari modelning moslashuvchanligini oshiradi va uning yanada murakkab masalalarni yechish imkoniyatini beradi. Python dasturlash tilida ushbu algoritmlarni oddiy va tushunarli tarzda amalga oshirish mumkin, bu esa ilmiy-tadqiqot va amaliy loyihalarda katta foyda keltiradi.

Foydalanilgan adabiyotlar

1. Bekmuratov Q.A. , Sun’iy intellekt va neyron tarmoqlar. O‘quv qo‘llanma, Samarqand – 2021.
2. Sadullayeva SH.A, Yusupov D.F., Yusupov F., Sun’iy intellect va neyronto‘rli texnologiyalar. O‘quv qo‘llanma, Urganch – 2021.
3. H.N.Zayniddinov, T.A.Xo‘jaqulov, M.P.Atadjanov, “Sun’iy intellekt” fanidan o‘quv qo‘llanma, Toshkent – 2018.
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
5. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.
6. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. [Online] Available at: https://www.deeplearningbook.org/
7. Интеллектуальные информационные системы и технологии: учебное пособие/ Ю.Ю. Громов, О.Г.Иванова, В.В. Алекссев и др. – тамбов: Издво ФГБОУ ВПО «ТГТУ», 2013.-244с.