



دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

# تمرین سوم هوش محاسباتی: شبکه های عصبی و کاربردها

## Neural Networks & Applications

نگارش

دانیال شفیعی

مهدی مهدیه

امیررضا نجفی

استاد راهنما

دکتر کارشناس

خرداد ۱۴۰۴

## فهرست مطالب

۰	مقدمه	۲
۱	مفاهیم و حل مسئله	۲
۲	کدزنی و پیاده سازی	۷

## ۰ مقدمه

هدف از این تمرین آشنایی بیشتر با شبکه های عصبی و استفاده ی بیشتر از آن ها در کاربردهای عملی است.

## ۱ مفاهیم و حل مسئله

۱. بله، هر نورون در یک شبکه عصبی حامل نوعی اطلاعات است؛ اما ماهیت و میزان «وضوح» این اطلاعات بسته به عمق لایه و ویژگی های بنیادین شبکه متفاوت است.
- چهار ویژگی بنیادی و سلسله مراتبی بودن نمایش:

## (آ) توابع غیرخطی (Nonlinearity)

- هر نورون پس از ترکیب خطی ورودی ها (ضرب وزن ها + بایاس) خروجی را از طریق تابعی مانند ReLU، sigmoid یا tanh عبور می دهد.
- بدون غیرخطی سازی، شبکه عملاً یک عملگر خطی بزرگ خواهد بود و قادر به تشخیص زیرویرگی های پیچیده نیست.
- تابع فعال سازی باعث می شود هر نورون تنها در صورت وقوع یک الگوی خاص «فعال» شود و در نتیجه به عنوان یک تشخیص دهنده ساده عمل کند.

## (ب) نمایش توزیع شده (Distributed Representation)

- برخلاف سیستم های سمبلیک که هر مفهوم را با یک واحد منفرد نمایش می دهند، شبکه های عصبی مفاهیم را به صورت همزمان در بردار فعال سازی تعداد زیادی نورون کدگذاری می کنند.
- این پراکندگی اطلاعات باعث افزایش مقاومت شبکه در برابر نویز و آسیب به نورون های منفرد می شود.
- هر نورون سهم جزئی اما معنادار در تشخیص زیرویرگی های ساده یا انتزاعی دارد.

## (ج) یادگیری گرادیان محور (Gradient-based Learning)

- با استفاده از الگوریتم پس انتشار (Backpropagation)، وزن ها و بایاس هر نورون به روزرسانی می شود تا خطای خروجی به کمترین مقدار برسد.
- در طی آموزش، هر نورون به زیرویرگی هایی پاسخ می دهد که برای کاهش خطا در مسئله مشخص مفیدند.
- در پایان آموزش، وزن های ورودی هر نورون تعیین می کنند که آن نورون به چه الگو یا ویژگی حساس باشد.

## (د) سلسله‌مراتب ویژگی‌ها (Hierarchical Feature Learning)

- لایه‌های ابتدایی شبکه‌های عمیق معمولاً به زیرویژگی‌های ساده مانند لبه‌های عمودی/افقی یا بافت‌ها حساس‌اند.
- لایه‌های میانی ترکیب این زیرویژگی‌ها را انجام داده و الگوهای پیچیده‌تر را می‌آموزند.
- در لایه خروجی (مثلاً نورون‌های softmax) احتمال تعلق هر ورودی به یک کلاس نهایی (مثلاً «گربه» یا «سگ») کدگذاری می‌شود.

۲. در شبکه‌های عصبی، «دانش» در قالب پارامترها (وزن‌ها و بایاس‌ها) ذخیره می‌شود و از طریق فرآیند آموزش شکل می‌گیرد؛ در ادامه، یک پاسخ یکپارچه و مرتب‌شده ارائه شده است:

## (آ) شکل‌گیری دانش در شبکه‌های عصبی

- تعریف ساختار شبکه (Architecture): انتخاب تعداد لایه‌ها (Input, Hidden, Output)، نوع آن‌ها (fully-connected، کانولوشن، بازگشتی و ...) و تعداد نورون در هر لایه.
- مقداردهی اولیه پارامترها (Initialization): وزن‌ها و بایاس‌ها معمولاً با توزیع‌های تصادفی (مثل Xavier یا He) مقداردهی می‌شوند.
- انتشار رو به جلو (Forward Propagation): برای هر ورودی  $x$ ، در هر لایه:

$$z^{(\ell)} = W^{(\ell)} a^{(\ell-1)} + b^{(\ell)}, \quad a^{(\ell)} = \sigma(z^{(\ell)})$$

در نهایت  $a^{(L)}$  خروجی نهایی شبکه است.

- محاسبه خطا (Loss Calculation): با تابع هزینه  $L(y_{\text{pred}}, y_{\text{true}})$ ، مانند MSE برای رگرسیون یا Cross-Entropy برای طبقه‌بندی.
- پس انتشار خطا (Backpropagation): مشتق تابع هزینه را نسبت به پارامترها محاسبه می‌کنیم:

$$\frac{\partial L}{\partial W^{(\ell)}}, \quad \frac{\partial L}{\partial b^{(\ell)}}$$

- به‌روزرسانی پارامترها (Optimization): با الگوریتم‌هایی مثل Gradient Descent یا Adam:

$$W^{(\ell)} \leftarrow W^{(\ell)} - \eta \frac{\partial L}{\partial W^{(\ell)}}, \quad b^{(\ell)} \leftarrow b^{(\ell)} - \eta \frac{\partial L}{\partial b^{(\ell)}}$$

این چرخه تا رسیدن به همگرایی تکرار می‌شود.

## (ب) فرمول‌بندی «معادل بودن» دو شبکه عصبی

- معادل تابعی (Exact Functional Equivalence) دو شبکه  $N(x) = f_{\theta_N}(x)$  و  $M(x) = f_{\theta_M}(x)$  دقیقاً معادل‌اند اگر:

$$\forall x \in X, \quad N(x) = M(x).$$

- معادل ساختاری تحت تبدیلات (Structural Equivalence) در لایه‌های Dense، جابجایی نورون‌ها (پرموتیشن  $\pi$ ) همراه با جابجایی سطرها/ستون‌های متناظر در  $W, b$ ، خروجی را تغییر نمی‌دهد.

iii. تقریب معادل (Approximate Equivalence) با فاصله خروجی  $d(x) = \|N(x) - M(x)\|_p$ :

$$\forall x \in X, d(x) < \epsilon \quad \text{یا} \quad \text{KL}(N(x) \| M(x)) < \delta.$$

### (ج) مثال ریاضی

i. حالت ساده خطی: دو شبکه خطی با یک لایه پنهان و  $\sigma(z) = z$ :

$$N(x) = W_2(W_1x + b_1) + b_2, \quad M(x) = W'_2(W'_1x + b'_1) + b'_2.$$

آن‌ها معادل‌اند اگر:

$$W_2W_1 = W'_2W'_1, \quad W_2b_1 + b_2 = W'_2b'_1 + b'_2.$$

ii. اشاره‌ای به حالت غیرخطی: در شبکه‌های غیرخطی (مثلاً ReLU)، تبدیلات پیچیده‌ترند؛ اما با ادغام BatchNorm یا تبدیلات جبری می‌توان مشابهت رفتار را نشان داد.

۳. در شبکه‌های عصبی، توانایی یادگیری، به‌خاطر سپاری (Memorization) و تعمیم (Generalization) بر پایه‌ی ساختار معماری، الگوریتم‌های آموزش و ویژگی‌های داده‌ها شکل می‌گیرد. در ادامه، این سه قابلیت را همراه با مبانی ریاضی و مثال‌های عینی بررسی می‌کنیم.

### (آ) یادگیری (Learning)

شبکه با کمینه‌سازی تابع هزینه

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N \ell(f_{\theta}(x_i), y_i)$$

و به‌کارگیری انتشار رو به جلو و پس‌انتشار خطا، (Backpropagation) پارامترهای  $\theta$  (وزن‌ها و بایاس‌ها) را با الگوریتم‌های گرادیان‌محور، (SGD) Adam و... به‌روزرسانی می‌کند:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} L(\theta).$$

- قضیه تقریب جهانی (Universal Approximation Theorem) هر شبکه‌ی با حداقل یک لایه پنهان و تابع فعال‌سازی غیرفابی (مثلاً ReLU یا Sigmoid) می‌تواند هر تابع پیوسته روی یک مجموعه‌ی کامپکت را تقریب بزند.

- ویژگی توزیعی (Distributed Representation) هر نورون یا زیرساختار فقط بخشی از ویژگی‌های داده را مدل می‌کند و با ترکیب میلیون‌ها پارامتر، شبکه قادر به نمایش الگوهای غیرخطی و سلسله‌مراتبی است.

### (ب) به‌خاطر سپاری (Memorization)

شبکه‌های overparameterized (پارامترها خیلی بزرگتر از نمونه‌ها) می‌توانند جزئیات حتی نویزی داده‌های آموزشی را حفظ کنند:

VC-Dimension Capacity: High و Complexity Rademacher بزرگ.

## ● مثال Bias–Variance Tradeoff:

Complexity  $\uparrow \rightarrow$  Bias  $\downarrow$ , Variance  $\uparrow$ , Memorization  $\uparrow$

اگر هیچ ضابطه‌ای (Regularization) وجود نداشته باشد، شبکه می‌تواند اطلاعات آموزشی را تقریباً کامل بازتولید کند.

## (ج) تعمیم (Generalization)

تعمیم یعنی عملکرد خوب روی داده‌های ندیده. این امر با ترکیب مکانیزم‌های explicit regularization و implicit regularization حاصل می‌شود: Inductive Bias

$$L_{\text{reg}}(\theta) = L(\theta) + \lambda \|\theta\|_2^2$$

● **Implicit Regularization**: رفتار SGD شبکه را به سمت مینیمم‌های تخت (flat minima) هدایت می‌کند.

● **Regularization صریح:**

– (L2) Weight Decay: افزودن  $\lambda \|\theta\|_2^2$  به تابع هزینه.

– Dropout: غیرفعال‌سازی تصادفی نورون‌ها.

– Early Stopping: پایان آموزش پیش از شروع شدید Overfitting.

● **Inductive Bias معماری:**

– CNN: اشتراک وزن‌ها و حساسیت به ویژگی‌های مکانی.

– RNN/Transformer: نگاشت توالی‌های زمانی و وابستگی‌های ترتیبی.

● **نرمال‌سازی (Batch Normalization)**: کاهش حساسیت به مقیاس وزن‌ها و تثبیت جریان گرادیان.

● **داده‌های متنوع و کافی**: کمیت و کیفیت داده‌های آموزشی پایه‌ی استخراج قاعده‌های عمومی و کاهش Overfitting است.

(د) پیوند با «معادل بودن دو شبکه» و «شکل‌گیری دانش»

شکل‌گیری دانش = پارامترهای بهینه  $\theta$  که از فرآیند یادگیری به دست می‌آیند. معادل بودن دو شبکه وقتی است که:

$$\forall x, N(x) = M(x) \quad \text{Functional (Exact)}$$

یا با پرموتیشن  $\pi$  روی نورون‌ها (Structural Symmetry)، یا تقریباً:

$$\|N(x) - M(x)\|_p < \varepsilon \quad \text{یا} \quad \text{KL}(N(x) \| M(x)) < \delta.$$

۴. در زیر سه نوع رایج از “توابع تبدیل نرونی” (یا به عبارت دیگر انواع نرون‌های مرتبه‌بالا و RBF) را به صورت ریاضی می‌بینید:

(آ) نرون درجه دوم (Quadratic Neuron):

$$\text{net}(\mathbf{x}) = \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j + \sum_{i=1}^n v_i x_i + b,$$

$$y = f(\text{net}(\mathbf{x})),$$

که در آن

- $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$
- ضرایب ضرب‌های درجه دوم،  $w_{ij}$
- ضرایب ترکیب خطی،  $v_i$
- بایاس،  $b$
- تابع فعال‌سازی،  $f(\dots)$ .

(ب) نرون کروی (Spherical / RBF Neuron):

$$\text{net}(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\mu}\| = \sqrt{\sum_{i=1}^n (x_i - \mu_i)^2}, \quad y = f(\text{net}(\mathbf{x})),$$

یا گونه‌ی مربعی بدون ریشه:

$$\text{net}(\mathbf{x}) = \sum_{i=1}^n (x_i - \mu_i)^2, \quad y = \exp(-\gamma \text{net}(\mathbf{x})),$$

که در آن

- $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$  مرکز نرون،
- $\gamma > 0$  ضریب پهنای باند،
- $f(\cdot)$  می‌تواند تابع خطی یا نمایی باشد.

(ج) نرون چندجمله‌ای (Polynomial Neuron): ابتدا ترکیب خطی و توان:

$$u = \sum_{i=1}^n w_i x_i + b, \quad y = u^d = \left( \sum_{i=1}^n w_i x_i + b \right)^d.$$

به صورت کلی برای ورودی چندبعدی:

$$y = \sum_{\alpha_1 + \dots + \alpha_n \leq d} w_{\alpha_1, \dots, \alpha_n} x_1^{\alpha_1} \cdots x_n^{\alpha_n},$$

که در آن

- $d$  درجه چندجمله‌ای،
- $\alpha_i \in \mathbb{N}_0$  اندیس‌های چندجمله‌ای،
- ضرایب متناظر  $w_{\alpha_1, \dots, \alpha_n}$ .

۵. سوال ۵

۶. (آ) طراحی پرسپترون تک‌لایه

فرض کنیم می‌خواهیم الگوها را به صورت برچسب  $t_1 = -1$  برای  $P_1$  و  $t_2 = +1$  برای  $P_2$  دسته‌بندی کنیم. باید  $w \in \mathbb{R}^3$  و بایاس  $b$  را طوری بیابیم که

$$\begin{cases} \text{sign}(w^\top P_1 + b) = -1, \\ \text{sign}(w^\top P_2 + b) = +1. \end{cases}$$

این معادلات به صورت نابرابری‌های زیر نوشته می‌شوند:

$$w^\top(-1, -1, 1) + b < 0, \quad w^\top(+1, -1, 1) + b > 0.$$

به سادگی می‌توانیم مثلاً وزن‌ها را به صورت  $w = (1, 0, 0)$ ، و بایاس  $b = 0$  انتخاب کنیم:

$$w^\top P_1 + b = -1 < 0, \quad w^\top P_2 + b = +1 > 0.$$

لذا تابع تصمیم  $y = \text{sign}(x_1)$  دو الگو را به درستی تفکیک می‌کند.

(ب) طراحی شبکه **Hamming**

شبکه همینگ برای  $N$  الگو  $P_k$  به صورت زیر است:

$$\mathbf{W} = \begin{bmatrix} P_1^\top \\ P_2^\top \end{bmatrix}, \quad y = \arg \max_k (\mathbf{W}x)_k.$$

برای  $P_1, P_2$  داریم:

$$\mathbf{W} = \begin{pmatrix} -1 & -1 & 1 \\ +1 & -1 & 1 \end{pmatrix}, \quad \text{انتخاب } k \text{ با بیشینه‌ی } \sum_i W_{k,i} x_i$$

(ج) طراحی شبکه **Hopfield**

شبکه هاپفیلد با الگوهای باینری  $\pm 1$  به کمک قاعده  $T = \sum_k P_k P_k^\top$  ساخته می‌شود. اینجا داریم:

$$T = P_1 P_1^\top + P_2 P_2^\top = \begin{pmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & -2 \\ 0 & -2 & 2 \end{pmatrix}.$$

سپس حالت نرونی‌ها با قاعده  $x_i \leftarrow \text{sign}(\sum_j T_{ij}x_j)$  به سمت نزدیک‌ترین الگو جذب می‌شود.

۷. (آ) طراحی مرز تصمیم و شبکه پرسپترون تک‌لایه

با انتخاب وزن‌ها و بایاس زیر:

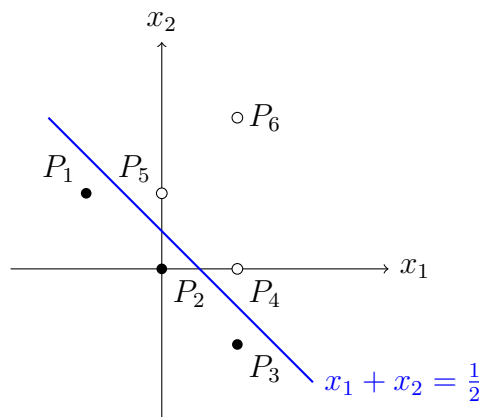
$$\mathbf{w} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad b = \frac{1}{2}$$

تابع فعال‌سازی گام به این صورت خواهد بود:

$$y = \begin{cases} 1, & \mathbf{w}^\top \mathbf{x} + b > 0, \\ 0, & \text{وگرنه.} \end{cases}$$

معادله مرز تصمیم:

$$-x_1 - x_2 + \frac{1}{2} = 0 \iff x_1 + x_2 = \frac{1}{2}.$$



(ب) تشخیص قابلیت جداسازی و تعیین بازه  $\varepsilon$

از نامعادلات زیر برای کلاس بندی استفاده می‌کنیم:

$$\begin{cases} -x_1 - x_2 + b > 0 & 1 \\ -x_1 - x_2 + b < 0 & 0 \end{cases}$$

نتیجه می‌شود که برای هر  $\varepsilon \geq 0$  می‌توان  $w_1 = w_2 = -1$  و  $b \in (0, 1)$  (مثلاً  $b = 1$ ) را انتخاب کرد و جداسازی خطی امکان‌پذیر است.

(ج) اجرای الگوریتم پرسپترون و نتایج نهایی

برای سه مقدار  $\varepsilon$  اجرای الگوریتم با نرخ یادگیری  $\eta = 1$ ، وزن و بایاس را از صفر مقداردهی کرده و تا خطای صفر تکرار می‌کنیم.

برنامه ۱: پیاده‌سازی الگوریتم پرسپترون



```

1 import numpy as np
2
3 def perceptron_train(P, t, lr=1, max_epochs=1000):
4     w = np.zeros(2)
5     b = 0.0
6     epc = 0
7     for epoch in range(max_epochs):
8         errors = 0
9         for x, target in zip(P, t):
10             y = 1 if np.dot(w, x) + b > 0 else 0
11             if y != target:
12                 errors += 1
13                 update = lr * (target - y)
14                 w += update * x
15                 b += update
16             if errors == 0:
17                 break
18         epc = epoch
19     return w, b, epc+1
20
21 epsilons = [1, 2, 6]
22 for eps in epsilons:
23     P = [
24         np.array([0,1]), np.array([1,-1]), np.array([-1,1]),
25         np.array([1,eps]), np.array([1,0]), np.array([0,0])
26     ]
27     t = [0,1,1,0,0,1]
28     w, b, epochs = perceptron_train(P, t)
29     print(f"={eps}: w={w}, b={b}, epochs={epochs}")

```

$\varepsilon = 1$ :  $w = (-1, -1)$ ,  $b = 1$ , epochs = 2,

$\varepsilon = 2$ :  $w = (-2, -2)$ ,  $b = 1$ , epochs = 4,

$\varepsilon = 6$ :  $w = (-3, -4)$ ,  $b = 3$ , epochs = 4.

#### (د) خلاصه نتایج

- مرز تصمیم:  $x_1 + x_2 = \frac{1}{2}$ .
- بازه  $\varepsilon$ :  $\varepsilon \geq 0$  (تمام مقادیر غیرمنفی).
- وزن‌ها و بایاس نهایی برای  $\varepsilon = 1, 2, 6$  مطابق جدول فوق.
- الگوریتم پرسپترون حداکثر تا ۴ دور همگرا شده و خطای صفر حاصل شد.

۸. سوال ۸

۹. سوال ۹

## ۲ کدزنی و پیاده سازی