# Short Questions

## ۱.۱

* **استفاده از تکنیک‌های رگولاریزیشن**

تکنیک‌هایی نظیر L2 Regularization (رگرسیون Ridge) یا Dropout که به صورت رندم خروجی برخی لایه‌ها را نادیده می‌گیرد، برای جلوگیری از Overfitشدن مدل استفاده می‌شوند. این تکنیک‌ها در واقع اجازه می‌دهند خطای ترین، در ازای Generalization بهتر افزایش یابد. از آن‌جایی که این تکنیک‌ها هنگام تست مدل با دیتای Validation اعمال نمی‌شوند، ممکن است در نتیجه خطای Validationمان کم‌تر باشد. در واقع، در اینجا مدل‌مان را Over-Regularize کرده‌ایم.

برای حل این مشکل،‌ می‌توان پارامترهای رگولاریزیشن نظیر نرخ Drop Out را طوری تغییر داد که دقت روی دیتای ترین، بیش از اندازه فدای دقت روی دیتای Validation نشود.

* **توزیع متفاوت دیتای Validation و ساده‌تر بودن طبقه‌بندی آن**

ممکن است دیتای Validation، از یک توزیع آماری متفاوت از دیتای ترین گردآوری شده‌باشد که طبقه‌بندی را ساده‌تر می‌کند. برای مثال،‌ دیتای Validation در نتیجه توزیع متفاوت، ساده‌تر باشد و تنها با چند فیچر بتوان خروجی را تخمین زد. در این حالت نیز ممکن است خطای Validation کم‌تر از خطای ترین باشد.

برای حل این مشکل، باید مطمئن شد که هم دیتای ترین، و هم دیتای Validation، به اندازه کافی بزرگ هستند و از یک توزیع یکسان نمونه‌برداری شده‌اند. با روش‌های آماری، می‌توان شاخص‌هایی نظیر میانگین، میانه، واریانس و ... را میان فیچرهای مختلف دیتای ترین و Validation محاسبه کرده، و از متعادل بودن این دو دیتاست اطمینان حاصل کرد.

## ۱.۲

اگر لایه‌های یک شبکه عصبی عمیق، تماما از توابع فعال‌سازی خطی استفاده می‌کردند، این شبکه تنها قادر به تشخیص رابطه خطی میان ورودی و خروجی بود؛ به عبارت دیگر، هر لایه صرفا یک نگاشت خطی از ورودی به خروجی بود، و در نتیجه کل شبکه را می‌شد با یک Perceptron تک‌لایه خطی مدل کرد.

بنابراین، اگر رابطه میان ورودی و خروجی یک رابطه غیر خطی است، لازم است که لایه‌های شبکه عصبی عمیق نیز غیر خطی باشند تا بتوان این روابط را Capture کرد.

## ۱.۳

به نظر می‌رسد که متد انتخاب شده بهینه نباشد، و چند مشکل با آن وجود دارد؛ نخست آنکه مقادیر انتخاب‌شده برای نرخ یادگیری، به نظر سلیقه‌ای می‌آیند و رابطه خاصی (چه خطی چه غیرخطی) بین آن‌ها نیست. همچنین برخی بازه‌های نسبتا بزرگ با این مقادیر انتخابی، نادیده گرفته می‌شوند. (مثلا بازه بین ۰.۲۱ تا ۰.۸۴، و احتمالا بازه بین ۰.۰۱ تا ۰.۱۶).

جدای از مقادیر در نظر گرفته‌شده، خود روش یافتن بهترین مقدار برای نرخ یادگیری نیز احتمالا بهینه نیست؛ یک کار بهتر،‌ تعریف یک دنباله بزرگ‌تر خطی در بازه معین‌شده، و ترین‌کردن کوتاه مدل روی هر یک از مقادیر این رنج است. منظور از ترین‌کردن کوتاه، محدود کردن تعداد Epoch هاست. با تعریف مناسب بازه میان مقادیر مختلف نرخ یادگیری، می‌توان نرخ یادگیری را در برابر خطای مدل پلات کرد، و نقطه مینیمم را، به عنوان نقطه بهینه نرخ یادگیری انتخاب نمود.

## ۱.۴

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۱.۵

لورم اپسیوم یک دسته‌خر فارسی است که.

# ‌Backpropagation

## **۲.۱**

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۲.۲

## ۲.۳

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۲.۴

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۲.۵

لورم اپسیوم یک دسته‌خر فارسی است که.

# Two Layer Neural Network

## ۳.۱

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۳.۲

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۳.۳

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۳.۴

لورم اپسیوم یک دسته‌خر فارسی است که.

# Hyperparameter Optimization

## ۴.۱

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۴.۲

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۴.۳

لورم اپسیوم یک دسته‌خر فارسی است که.

# Decision Tree

## ۵.۱

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۵.۲

لورم اپسیوم یک دسته‌خر فارسی است که.

# AdaBoost Algorithm

## ۶.۱

لورم اپسیوم یک دسته‌خر فارسی است که.

## ۶.۲

لورم اپسیوم یک دسته‌خر فارسی است که.