بسم الله الرحمن الرحیم

*دانشکده فنی حرفه ای آیت الله خامنه *

تمرین : بخش اول: 20/12/1 Machine Learning

درس: مباحث ویژه

استاد: محمد احمد زاده

اعضای گروه: سمیرا صالحی. سمانه بهاری

بخش 1: 20/12/1403 Machine Learning

A. Supervised Learning و Unsupervised Learning چه تفاوتی دارند؟

تفاوت اصلی بین یادگیری نظارت‌شده (Supervised Learning) و یادگیری بدون نظارت (Unsupervised Learning) در نحوه آموزش مدل و نوع داده‌هایی است که در اختیار مدل قرار می‌گیرد:

یادگیری نظارت‌شده (Supervised Learning):

* **داده‌ها:** در این روش، مدل با استفاده از داده‌های "برچسب‌دار" آموزش داده می‌شود. یعنی، هر نمونه از داده‌ها دارای یک ورودی (features) و یک خروجی (label) مشخص است. هدف مدل، یادگیری یک تابع است که بتواند ورودی را به خروجیِ صحیح نگاشت کند.
* **هدف:** پیش‌بینی خروجی (label) برای ورودی‌های جدید و ناشناخته.
* **مثال‌ها:**
  * **Classification:** دسته‌بندی ایمیل‌ها به عنوان "spam" یا "not spam".
  * **Regression:** پیش‌بینی قیمت خانه بر اساس ویژگی‌های مختلف (متراژ، موقعیت و غیره).
* **الگوریتم‌های رایج:**
  * رگرسیون خطی (Linear Regression)
  * رگرسیون لجستیک (Logistic Regression)
  * ماشین بردار پشتیبان (Support Vector Machine – SVM)
  * درخت تصمیم (Decision Tree)
  * جنگل تصادفی (Random Forest)
  * شبکه‌های عصبی (Neural Networks)

یادگیری بدون نظارت (Unsupervised Learning):

* **داده‌ها:** در این روش، مدل با استفاده از داده‌های "بدون برچسب" آموزش داده می‌شود. یعنی، داده‌ها فقط شامل ورودی (features) هستند و هیچ خروجی (label) از پیش تعیین‌شده‌ای وجود ندارد.
* **هدف:** کشف الگوها، ساختارها، و روابط پنهان در داده‌ها.
* **مثال‌ها:**
  * **Clustering:** گروه‌بندی مشتریان بر اساس رفتار خرید آن‌ها.
  * **Dimensionality Reduction:** کاهش تعداد ویژگی‌های یک مجموعه داده در حالی که اطلاعات مهم حفظ شود.
  * **Association Rule Mining:** پیدا کردن ارتباط بین آیتم‌های مختلف در یک تراکنش (مانند تحلیل سبد خرید).
* **الگوریتم‌های رایج:**
  * K-Means Clustering

* Hierarchical Clustering
* PCA (Principal Component Analysis)
* t-SNE (t-distributed Stochastic Neighbor Embedding)
* Anomaly Detection

خلاصه تفاوت‌ها:

ویژگی	یادگیری نظارت‌شده (Supervised Learning)	یادگیری بدون نظارت (Unsupervised Learning)
نوع داده‌ها	برچسب‌دار (Labeled)	بدون برچسب (Unlabeled)
هدف	پیش‌بینی خروجی	کشف الگوها و ساختارها
نوع مسائل	Classification, Regression	Clustering, Dimensionality Reduction, Association Rule Mining

به طور خلاصه، اگر هدف شما پیش‌بینی یک خروجی مشخص بر اساس ورودی‌ها باشد و داده‌های برچسب‌دار در اختیار دارید، باید از یادگیری نظارت‌شده استفاده کنید. اگر هدف شما کشف الگوها و ساختارها در داده‌ها باشد و داده‌های بدون برچسب در اختیار دارید، باید از یادگیری بدون نظارت استفاده کنید.

B. چرا Feature Scaling در الگوریتم‌های Machine Learning ضروری است؟

Feature Scaling در الگوریتم‌های Machine Learning به دلایل مختلفی ضروری است که می‌توان آن‌ها را به صورت زیر دسته‌بندی کرد:

1. تأثیر مقیاس متغیرها بر عملکرد الگوریتم:

* **الگوریتم‌های مبتنی بر فاصله:** الگوریتم‌هایی مانند K-Nearest Neighbors (KNN)، Support Vector Machines (SVM)، و K-Means Clustering به شدت تحت تأثیر مقیاس متغیرها هستند. اگر یک ویژگی دارای مقادیر بسیار بزرگتری نسبت به سایر ویژگی‌ها باشد، فاصله بین نقاط داده بیشتر توسط آن ویژگی تعیین می‌شود و سایر ویژگی‌ها نادیده گرفته می‌شوند. این امر می‌تواند منجر به نتایج نادرست و عملکرد ضعیف مدل شود.

* **الگوریتم‌های مبتنی بر گرادیان:** در الگوریتم‌هایی مانند رگرسیون خطی (Linear Regression) و شبکه‌های عصبی (Neural Networks) که از گرادیان کاهشی (Gradient Descent) برای بهینه‌سازی استفاده می‌کنند، مقیاس متغیرها می‌تواند بر سرعت و پایداری فرآیند یادگیری تأثیر بگذارد. اگر مقیاس متغیرها متفاوت باشد، ممکن است گرادیان در جهت‌های مختلف به سرعت تغییر کند و الگوریتم به کندی همگرا شود یا اصلاً همگرا نشود.

2. بهبود تفسیرپذیری مدل:

- Feature Scaling می‌تواند تفسیرپذیری مدل را بهبود بخشد، به ویژه در مدل‌های خطی مانند رگرسیون خطی. وقتی متغیرها در یک مقیاس مشابه قرار دارند، ضرایب رگرسیون (Regression Coefficients) به راحتی قابل مقایسه هستند و می‌توان فهمید که کدام ویژگی‌ها تأثیر بیشتری بر پیش‌بینی دارند.

3. جلوگیری از سرریز و کمبود محاسباتی:

- در برخی موارد، مقادیر بسیار بزرگ یا بسیار کوچک در ویژگی‌ها می‌توانند منجر به مشکلات محاسباتی مانند سرریز (Overflow) یا کمبود (Underflow) شوند. Feature Scaling می‌تواند از این مشکلات جلوگیری کند و پایداری محاسبات را بهبود بخشد.

چه زمانی Feature Scaling ضروری نیست؟

- الگوریتم‌های مبتنی بر درخت: الگوریتم‌هایی مانند درخت تصمیم (Decision Tree) و جنگل تصادفی (Random Forest) به Feature Scaling حساس نیستند. این الگوریتم‌ها بر اساس تقسیم‌بندی داده‌ها بر اساس مقادیر ویژگی‌ها عمل می‌کنند و مقیاس متغیرها تأثیری بر این فرآیند ندارد.

روش‌های رایج Feature Scaling:

* **Standardization (Z-score Normalization):** مقادیر را به گونه‌ای تغییر مقیاس می‌دهد که میانگین برابر با صفر و انحراف معیار برابر با یک شود.
  * فرمول: `(x – mean) / standard_deviation`
* **Min-Max Scaling:** مقادیر را به بازه [0, 1] تغییر مقیاس می‌دهد.
  * فرمول: `(x – min) / (max – min)`
* **Robust Scaling:** مشابه Standardization است، اما از میانه (Median) و محدوده بین چارکی (Interquartile Range – IQR) به جای میانگین و انحراف معیار استفاده می‌کند. این روش در برابر داده‌های پرت (Outliers) مقاوم‌تر است.
* **Normalization:** مقادیر را به گونه‌ای تغییر مقیاس می‌دهد که طول بردار برابر با یک شود.

نکته مهم:

- هنگام استفاده از Feature Scaling، مهم است که Scaling را فقط بر روی داده‌های آموزشی (Training Data) اعمال کنید و سپس از همان Scaling برای داده‌های اعتبارسنجی (Validation

Data) و تست (Test Data) استفاده کنید. این کار از ورود اطلاعات از داده‌های اعتبارسنجی و تست به فرآیند آموزش جلوگیری می‌کند و از overfitting جلوگیری می‌کند.

در مجموع، Feature Scaling یک گام مهم در پیش‌پردازش داده‌ها است که می‌تواند به طور قابل توجهی عملکرد و پایداری الگوریتم‌های Machine Learning را بهبود بخشد. انتخاب روش مناسب Feature Scaling بستگی به نوع داده‌ها و الگوریتم مورد استفاده دارد.

C. Standardization و Normalization چه تفاوتی دارند؟

Standardization و Normalization هر دو تکنیک‌های Feature Scaling هستند که برای تغییر مقیاس متغیرها (features) در داده‌ها استفاده می‌شوند، اما روش کار و نتایج متفاوتی دارند. درک این تفاوت‌ها برای انتخاب روش مناسب در یک مسئله خاص مهم است.

:Standardization (Z-score Normalization)

* **هدف:** تغییر مقیاس داده‌ها به گونه‌ای که میانگین آن‌ها برابر با 0 و انحراف معیار آن‌ها برابر با 1 شود. به عبارت دیگر، داده‌ها حول میانگین خود متمرکز می‌شوند و پراکندگی آن‌ها بر اساس انحراف معیار تنظیم می‌شود.
* **فرمول:**

```
X_scaled = (x – mean) / standard_deviation
```

که در آن:

* `x` مقدار اصلی ویژگی است.
* `mean` میانگین ویژگی است.
* `standard_deviation` انحراف معیار ویژگی است.

* **ویژگی‌ها:**
* داده‌ها لزوماً در بازه خاصی قرار نمی‌گیرند. ممکن است مقادیر منفی و مثبت بزرگتر از 1 داشته باشند.
* کمک می‌کند تا داده‌ها توزیع نرمال‌تری داشته باشند (البته تضمینی نیست).
* به خوبی با داده‌های پرت (outliers) کار می‌کند، زیرا انحراف معیار به طور کلی تحت تأثیر داده‌های پرت قرار نمی‌گیرد.
* برای الگوریتم‌هایی که فرض می‌کنند داده‌ها توزیع نرمال دارند (مانند رگرسیون خطی)، مناسب است.

* **چه زمانی استفاده کنیم؟**
  * وقتی الگوریتم شما فرض می‌کند داده‌ها توزیع نرمال دارند.
  * وقتی می‌خواهید داده‌ها را حول میانگین خود متمرکز کنید.
  * وقتی داده‌های پرت دارید و می‌خواهید تأثیر آن‌ها را کاهش دهید.

**Normalization (Min-Max Scaling):*

که در آن:

* `x` مقدار اصلی ویژگی است.
* `min` حداقل مقدار ویژگی است.
* `max` حداکثر مقدار ویژگی است.

* **ویژگی‌ها:**
  * داده‌ها حتماً در بازه [0, 1] قرار می‌گیرند.
  * توزیع داده‌ها را تغییر نمی‌دهد.
  * به شدت تحت تأثیر داده‌های پرت قرار می‌گیرد. اگر داده‌های پرت زیادی داشته باشید، ممکن است بازه [0, 1] به شدت فشرده شود و اطلاعات مفید از دست برود.
  * برای الگوریتم‌هایی که به بازه خاصی از مقادیر نیاز دارند، مناسب است.
* **چه زمانی استفاده کنیم؟**
  * وقتی می‌خواهید داده‌ها را در بازه مشخصی (مثلاً [0, 1]) قرار دهید.
  * وقتی توزیع داده‌ها نرمال نیست و می‌خواهید آن را حفظ کنید.
  * وقتی داده‌های پرت زیادی ندارید.
  * برای الگوریتم‌هایی مانند شبکه‌های عصبی که معمولاً به ورودی‌هایی در بازه [0, 1] نیاز دارند، مناسب است.

خلاصه تفاوت‌ها:

ویژگی	Normalization	Standardization
هدف	میانگین 0، انحراف معیار 1	بازه [0, 1]
بازه مقادیر	بدون بازه مشخص	[0, 1]
تأثیر روی توزیع	تلاش برای نرمال کردن توزیع	حفظ توزیع اصلی

حساسیت به پرت	کمتر حساس	بیشتر حساس
کاربرد	توزیع نرمال، کاهش تأثیر پرت‌ها	بازه مشخص، حفظ توزیع اصلی

انتخاب روش مناسب:

انتخاب بین Standardization و Normalization بستگی به نوع داده‌ها و الگوریتم مورد استفاده دارد.

* اگر داده‌های پرت زیادی دارید، Standardization معمولاً گزینه بهتری است.
* اگر الگوریتم شما فرض می‌کند داده‌ها توزیع نرمال دارند، Standardization مناسب است.
* اگر می‌خواهید داده‌ها را در بازه مشخصی قرار دهید، Normalization مناسب است.
* اگر توزیع داده‌ها نرمال نیست و می‌خواهید آن را حفظ کنید، Normalization را انتخاب کنید.

در نهایت، بهترین راه برای انتخاب روش مناسب، آزمایش هر دو روش و مقایسه عملکرد مدل است.

D. چرا Min-Max Normalization برای مقیاس‌بندی داده‌ها استفاده می‌شود؟

Min-Max Normalization به دلایل مختلفی برای مقیاس‌بندی داده‌ها مورد استفاده قرار می‌گیرد:

1. مقیاس‌بندی داده‌ها در یک بازه مشخص:

- **تضمین بازه [0, 1]:** مهم‌ترین دلیل استفاده از Min-Max Normalization این است که داده‌ها را در بازه [0, 1] قرار می‌دهد. این ویژگی برای بسیاری از الگوریتم‌های یادگیری ماشین مفید است، به خصوص الگوریتم‌هایی که انتظار دارند ورودی‌ها در یک بازه خاص قرار داشته باشند.

2. ساده‌سازی محاسبات و بهبود عملکرد:

* **بهبود عملکرد الگوریتم‌ها:** برخی از الگوریتم‌ها، مانند شبکه‌های عصبی، در صورتی که ورودی‌ها در بازه [0, 1] باشند، عملکرد بهتری دارند. این مقیاس‌بندی می‌تواند فرآیند یادگیری را تسریع کند و از مشکلات مربوط به اعداد بزرگ جلوگیری کند.

* **محاسبات آسان‌تر:** عملیات ریاضی با اعدادی که در بازه [0, 1] قرار دارند، معمولاً ساده‌تر و سریع‌تر هستند.

3. تفسیرپذیری بهتر:

* **مقایسه آسان‌تر:** وقتی تمام ویژگی‌ها در بازه [0, 1] قرار دارند، مقایسه اهمیت نسبی آن‌ها آسان‌تر می‌شود. اگر یک ویژگی مقدار بالایی در این بازه داشته باشد، نشان‌دهنده اهمیت بیشتر آن در مقایسه با ویژگی‌هایی است که مقادیر پایین‌تری دارند.
* **درک آسان‌تر:** مقادیر بین 0 و 1 معمولاً برای انسان قابل فهم‌تر هستند و درک بهتری از داده‌ها فراهم می‌کنند.

4. استفاده در الگوریتم‌های خاص:

* **شبکه‌های عصبی:** همانطور که اشاره شد، شبکه‌های عصبی اغلب به ورودی‌هایی در بازه [0, 1] یا [-1, 1] نیاز دارند. Min-Max Normalization یک روش رایج برای آماده‌سازی داده‌ها برای این الگوریتم‌ها است.
* **الگوریتم‌های مبتنی بر فاصله:** در حالی که Standardization معمولاً برای الگوریتم‌های مبتنی بر فاصله مانند KNN توصیه می‌شود، Min-Max Normalization نیز می‌تواند در این الگوریتم‌ها استفاده شود، به خصوص اگر مقادیر ویژگی‌ها باید در یک بازه مشخص قرار داشته باشند.

5. حفظ شکل توزیع داده‌ها:

- **عدم تغییر توزیع:** برخلاف Standardization که سعی در تغییر توزیع داده‌ها به توزیع نرمال دارد، Min-Max Normalization شکل توزیع اصلی داده‌ها را حفظ می‌کند. این ویژگی زمانی مهم است که می‌خواهید ویژگی‌های آماری داده‌ها را تا حد امکان حفظ کنید.

معایب Min-Max Normalization:

* **حساسیت به داده‌های پرت:** یکی از بزرگترین معایب Min-Max Normalization این است که به شدت تحت تأثیر داده‌های پرت (outliers) قرار می‌گیرد. اگر داده‌های پرت زیادی در مجموعه داده وجود داشته باشد، مقیاس‌بندی Min-Max می‌تواند بازه [0, 1] را به شدت فشرده کند و اطلاعات مفید از دست برود.
* **محدودیت در بازه [0, 1]:** گاهی اوقات، محدود کردن داده‌ها به بازه [0, 1] ممکن است مناسب نباشد. در برخی موارد، ممکن است نیاز باشد که داده‌ها در یک بازه بزرگتر یا کوچکتر قرار گیرند.

جایگزین‌ها:

* **Standardization:** اگر داده‌های پرت زیادی دارید، Standardization (Z-score Normalization) معمولاً گزینه بهتری است.
* **Robust Scaling:** این روش از میانه و محدوده بین چارکی (IQR) به جای میانگین و انحراف معیار استفاده می‌کند و در برابر داده‌های پرت مقاوم‌تر است.

در نهایت، انتخاب بین Min-Max Normalization و سایر روش‌های مقیاس‌بندی بستگی به نوع داده‌ها، الگوریتم مورد استفاده، و اهداف شما دارد. اگر می‌خواهید داده‌ها را در بازه [0, 1] قرار دهید، توزیع داده‌ها را حفظ کنید، و داده‌های پرت زیادی ندارید، Min-Max Normalization یک گزینه مناسب است.

E. Z-Score Normalization چیست و چرا کاربرد دارد؟

Z-score normalization، که با نام standardization هم شناخته می‌شود، یک تکنیک مقیاس‌بندی داده (feature scaling) است که در یادگیری ماشین و آمار استفاده می‌شود. هدف اصلی این روش، تبدیل داده‌ها به گونه‌ای است که میانگین (mean) برابر با صفر و انحراف معیار (standard deviation) برابر با یک شود.

فرمول Z-score:

برای هر مقدار داده (`x`)، Z-score با استفاده از فرمول زیر محاسبه می‌شود:

```
Z = (x – μ) / σ
```

که در آن:

* `x`: مقدار داده‌ای که می‌خواهیم نرمال‌سازی کنیم.
* `μ`: میانگین مجموعه داده‌ای که `x` به آن تعلق دارد.
* `σ`: انحراف معیار مجموعه داده‌ای که `x` به آن تعلق دارد.
* `z`: مقدار Z-score محاسبه شده برای `x`.

چرا Z-score normalization کاربرد دارد؟

1. **مقایسه داده‌ها با مقیاس‌های مختلف:**

* وقتی داده‌ها از منابع مختلف با مقیاس‌های متفاوت جمع‌آوری شده‌اند، Z-score normalization به ما اجازه می‌دهد تا آن‌ها را با هم مقایسه کنیم. به عنوان مثال، فرض کنید دو ویژگی داریم: یکی با مقادیر بین 0 تا 1000 و دیگری با مقادیر بین 0 تا 1. بدون نرمال‌سازی، مقایسه این دو ویژگی دشوار است. Z-score normalization این مشکل را حل می‌کند.

2. **بهبود عملکرد الگوریتم‌های یادگیری ماشین:**

* بسیاری از الگوریتم‌های یادگیری ماشین، مانند رگرسیون خطی (Linear Regression)، رگرسیون لجستیک (Logistic Regression)، و شبکه‌های عصبی (Neural Networks)، عملکرد بهتری دارند وقتی که داده‌ها نرمال‌سازی شده باشند. این به این دلیل است که الگوریتم‌ها می‌توانند سریع‌تر و پایدارتر به جواب بهینه برسند.

* الگوریتم‌هایی که از فاصله (distance) برای اندازه‌گیری شباهت استفاده می‌کنند، مانند K-Nearest Neighbors (KNN) و K-Means clustering، به شدت تحت تأثیر مقیاس متغیرها هستند. Z-score normalization اطمینان می‌دهد که تمام ویژگی‌ها به یک اندازه در محاسبات فاصله نقش دارند.

3. **حساسیت کمتر به مقادیر پرت (Outliers):**

* Z-score normalization در مقایسه با Min-Max scaling، کمتر تحت تأثیر مقادیر پرت قرار می‌گیرد. اگرچه مقادیر پرت می‌توانند بر میانگین و انحراف معیار تأثیر بگذارند، اما تأثیر آن‌ها بر Z-score به اندازه Min-Max scaling نیست.

4. **توزیع نرمال (Normal Distribution):**

- Z-score normalization به تبدیل داده‌ها به توزیع نرمال کمک می‌کند. بسیاری از الگوریتم‌های آماری و یادگیری ماشین فرض می‌کنند که داده‌ها توزیع نرمال دارند. اگرچه Z-score normalization تضمین نمی‌کند که داده‌ها کاملاً نرمال شوند، اما آن‌ها را به توزیع نرمال نزدیک‌تر می‌کند.

چه زمانی از Z-score normalization استفاده کنیم؟

* وقتی داده‌ها از منابع مختلف با مقیاس‌های متفاوت جمع‌آوری شده‌اند.
* وقتی از الگوریتم‌های یادگیری ماشینی استفاده می‌کنید که به نرمال‌سازی داده‌ها حساس هستند.
* وقتی می‌خواهید تأثیر مقادیر پرت را کاهش دهید.

* وقتی می‌خواهید داده‌ها را به توزیع نرمال نزدیک‌تر کنید.

مثال:

فرض کنید دو دانش‌آموز در دو آزمون مختلف شرکت کرده‌اند. نمره دانش‌آموز اول در آزمون A برابر با 80 است و نمره دانش‌آموز دوم در آزمون B برابر با 90 است. برای اینکه بتوانیم عملکرد این دو دانش‌آموز را با هم مقایسه کنیم، باید نمرات آن‌ها را نرمال‌سازی کنیم.

* آزمون A: میانگین = 70، انحراف معیار = 10
* آزمون B: میانگین = 80، انحراف معیار = 5

Z-score برای دانش‌آموز اول:

```
Z1 = (80 – 70) / 10 = 1
```

Z-score برای دانش‌آموز دوم:

```
Z2 = (90 – 80) / 5 = 2
```

با توجه به Z-score، دانش‌آموز دوم عملکرد بهتری نسبت به دانش‌آموز اول داشته است، زیرا Z-score او بزرگتر است.

به طور خلاصه، Z-score normalization یک ابزار قدرتمند برای پیش‌پردازش داده‌ها است که می‌تواند به بهبود عملکرد الگوریتم‌های یادگیری ماشین و تفسیرپذیری داده‌ها کمک کند.

F. Regularization در الگوریتم‌های Machine Learning چیست؟

Regularization (منظم‌سازی) یک تکنیک مهم در الگوریتم‌های یادگیری ماشین است که برای جلوگیری از Overfitting (بیش‌برازش) استفاده می‌شود. Overfitting زمانی رخ می‌دهد که مدل به جای یادگیری الگوهای اصلی در داده‌ها، جزئیات و نویزهای موجود در داده‌های آموزشی را نیز یاد می‌گیرد. این باعث می‌شود که مدل روی داده‌های آموزشی عملکرد خوبی داشته باشد، اما روی داده‌های جدید (تست) عملکرد ضعیفی داشته باشد.

هدف Regularization:

هدف اصلی Regularization، ساده‌سازی مدل است به گونه‌ای که مدل فقط الگوهای مهم و کلی را یاد بگیرد و از یادگیری نویزها و جزئیات غیرضروری پرهیز کند. این کار باعث می‌شود که مدل روی داده‌های جدید بهتر عمل کند (Generalization).

چگونه Regularization کار می‌کند؟

Regularization با افزودن یک "penalty" (جریمه) به تابع هزینه (cost function) مدل کار می‌کند. این جریمه متناسب با پیچیدگی مدل است. مدل‌های پیچیده (که معمولاً دارای وزن‌های بزرگتر هستند) جریمه بیشتری دریافت می‌کنند. هدف این است که مدل را مجبور کنیم تا وزن‌های خود را کوچک نگه دارد، که این به ساده‌سازی مدل و جلوگیری از overfitting کمک می‌کند.

انواع اصلی Regularization:

1. **L1 Regularization (Lasso Regression):**

- به تابع هزینه، مجموع قدر مطلق وزن‌ها اضافه می‌شود:
```
|Cost = Loss + λ * Σ |wi
```
که در آن:
  * `Loss` تابع هزینه اصلی (مانند Mean Squared Error برای رگرسیون) است.
  * `λ` (لامبدا) پارامتر Regularization است که میزان جریمه را کنترل می‌کند.
  * `wi` وزن‌های مدل هستند.
* **ویژگی‌ها:**
  * L1 Regularization می‌تواند وزن برخی از ویژگی‌ها را به صفر برساند (feature selection). این ویژگی باعث می‌شود که مدل ساده‌تر و قابل تفسیرتر شود.
  * مناسب برای مدل‌هایی که تعداد زیادی ویژگی دارند و می‌خواهیم ویژگی‌های مهم را انتخاب کنیم.

2. **L2 Regularization (Ridge Regression):**

- به تابع هزینه، مجموع مربعات وزن‌ها اضافه می‌شود:
```
Cost = Loss + λ * Σ wi^2
```
* **ویژگی‌ها:**
  * L2 Regularization وزن‌ها را به صفر نزدیک می‌کند، اما معمولاً آن‌ها را صفر نمی‌کند.
  * وزن‌ها را به طور یکنواخت کاهش می‌دهد.
  * مناسب برای مدل‌هایی که تمام ویژگی‌ها به نوعی مهم هستند.

3. **Elastic Net Regularization:**

- ترکیبی از L1 و L2 Regularization است:
```
Cost = Loss + λ1 * Σ |wi| + λ2 * Σ wi^2
```
که در آن:
* `λ1` و `λ2` پارامترهای Regularization هستند که میزان جریمه L1 و L2 را کنترل می‌کنند.
* **ویژگی‌ها:**
  * Elastic Net مزایای هر دو روش L1 و L2 را دارد.
  * می‌تواند ویژگی‌های غیرضروری را حذف کند و همچنین وزن‌ها را به طور یکنواخت کاهش دهد.
  * مناسب برای مواردی که نمی‌دانیم کدام یک از L1 یا L2 بهتر است.

نکات مهم:

* **انتخاب مقدار λ (پارامتر Regularization):** مقدار λ باید با استفاده از تکنیک‌هایی مانند Cross-Validation (اعتبارسنجی متقابل) انتخاب شود. مقدار مناسب λ بستگی به مجموعه داده و مدل دارد.
* **مقیاس‌بندی داده‌ها:** قبل از استفاده از Regularization، مهم است که داده‌ها را مقیاس‌بندی کنید (مانند Z-score normalization یا Min-Max scaling). این کار باعث می‌شود که Regularization به طور یکنواخت روی تمام ویژگی‌ها تأثیر بگذارد.

خلاصه:

Regularization یک تکنیک قدرتمند برای جلوگیری از overfitting و بهبود عملکرد مدل‌های یادگیری ماشین است. با افزودن یک جریمه به تابع هزینه، Regularization مدل را مجبور می‌کند تا وزن‌های خود را کوچک نگه دارد و از یادگیری نویزها و جزئیات غیرضروری پرهیز کند. انتخاب نوع مناسب Regularization (L1، L2، یا Elastic Net) و مقدار مناسب λ بستگی به مجموعه داده و مدل دارد.

G. Overfitting و Underfitting چه مشکلاتی را در Model-building به وجود می‌آورند؟

Overfitting و Underfitting دو مشکل رایج در فرآیند ساخت مدل‌های یادگیری ماشین هستند که می‌توانند عملکرد مدل را به شدت تحت تأثیر قرار دهند. درک این مفاهیم و نحوه مقابله با آن‌ها برای ساخت مدل‌های دقیق و قابل اعتماد ضروری است.

1. Overfitting (بیش‌برازش)

تعریف:
Overfitting زمانی رخ می‌دهد که مدل به جای یادگیری الگوهای کلی و اصلی در داده‌ها، جزئیات و نویزهای موجود در داده‌های آموزشی را نیز یاد می‌گیرد. این باعث می‌شود که مدل روی داده‌های آموزشی عملکرد بسیار خوبی داشته باشد، اما روی داده‌های جدید (تست) عملکرد ضعیفی از خود نشان دهد.

مشکلات ناشی از Overfitting:
- **عملکرد ضعیف روی داده‌های جدید:** مدل قادر به تعمیم (Generalization) نیست و نمی‌تواند الگوهای کلی را تشخیص دهد.
- **حساسیت به نویز:** مدل به نویزها و جزئیات غیرضروری در داده‌های آموزشی حساس می‌شود و این باعث کاهش دقت آن روی داده‌های تست می‌گردد.
- **پیچیدگی بیش از حد مدل:** مدل‌های بیش‌برازش شده معمولاً بسیار پیچیده هستند و پارامترهای زیادی دارند که این موضوع باعث افزایش زمان آموزش و پیش‌بینی می‌شود.

راه‌حل‌های Overfitting:
- **استفاده از Regularization:** تکنیک‌هایی مانند L1 و L2 Regularization به ساده‌سازی مدل و کاهش پیچیدگی آن کمک می‌کنند.
- **افزایش حجم داده‌های آموزشی:** با افزایش تعداد داده‌ها، مدل فرصت بیشتری برای یادگیری الگوهای کلی دارد.
- **کاهش پیچیدگی مدل:** کاهش تعداد لایه‌ها در شبکه‌های عصبی یا کاهش تعداد ویژگی‌ها در مدل‌های دیگر می‌تواند به جلوگیری از Overfitting کمک کند.
- **استفاده از Dropout:** در شبکه‌های عصبی، Dropout می‌تواند با غیرفعال کردن تصادفی برخی از نورون‌ها از Overfitting جلوگیری کند.
- **اعتبارسنجی متقابل (Cross-Validation):** این روش به ارزیابی بهتر عملکرد مدل روی داده‌های جدید کمک می‌کند.

---

2. Underfitting (کم‌برازش)

تعریف:
Underfitting زمانی رخ می‌دهد که مدل قادر به یادگیری الگوهای موجود در داده‌های آموزشی نیست. این باعث می‌شود که مدل هم روی داده‌های آموزشی و هم روی داده‌های جدید عملکرد ضعیفی داشته باشد.

مشکلات ناشی از Underfitting:
- **عملکرد ضعیف روی داده‌های آموزشی و تست:** مدل قادر به تشخیص الگوهای اصلی نیست و دقت پایینی دارد.
- **ساده‌بودن بیش از حد مدل:** مدل‌های کم‌برازش شده معمولاً بسیار ساده هستند و نمی‌توانند پیچیدگی داده‌ها را درک کنند.
- **عدم توانایی در تعمیم:** مدل حتی روی داده‌های آموزشی نیز عملکرد ضعیفی دارد و نمی‌تواند الگوها را به درستی یاد بگیرد.

راه‌حل‌های Underfitting:
- **افزایش پیچیدگی مدل:** اضافه کردن لایه‌های بیشتر در شبکه‌های عصبی یا استفاده از مدل‌های پیچیده‌تر می‌تواند به بهبود یادگیری مدل کمک کند.
- **استفاده از ویژگی‌های بیشتر:** اضافه کردن ویژگی‌های مرتبط به مدل می‌تواند به یادگیری الگوهای پیچیده‌تر کمک کند.
- **کاهش Regularization:** اگر از Regularization استفاده می‌کنید، کاهش مقدار آن می‌تواند به بهبود یادگیری مدل کمک کند.
- **افزایش زمان آموزش:** در برخی موارد، افزایش تعداد epochها در آموزش مدل می‌تواند به بهبود یادگیری کمک کند.

تفاوت Overfitting و Underfitting

ویژگی	Underfitting	Overfitting
عملکرد روی داده‌های آموزشی	بسیار خوب	ضعیف
عملکرد روی داده‌های تست	ضعیف	ضعیف
پیچیدگی مدل	بسیار پیچیده	بسیار ساده
علت اصلی	یادگیری نویزها و جزئیات غیرضروری	عدم یادگیری الگوهای اصلی
راه‌حل‌ها	Regularization، افزایش داده‌ها، Dropout	افزایش پیچیدگی مدل، کاهش Regularization

---

جمع‌بندی

Overfitting و Underfitting دو چالش اصلی در ساخت مدل‌های یادگیری ماشین هستند. Overfitting باعث می‌شود مدل روی داده‌های آموزشی بیش از حد دقیق باشد اما روی داده‌های جدید عملکرد ضعیفی داشته باشد، در حالی که Underfitting باعث می‌شود مدل حتی روی داده‌های آموزشی نیز عملکرد ضعیفی داشته باشد. برای مقابله با این مشکلات، باید تعادل مناسبی بین پیچیدگی مدل و حجم داده‌ها برقرار کرد و از تکنیک‌هایی مانند Regularization، Cross-Validation و Dropout استفاده نمود.

H. Cross-Validation چرا در Train/Test Split کاربرد دارد؟

(تایید متقابل) یک تکنیک مهم در فرآیند ارزیابی مدل‌های یادگیری ماشین است که به منظور افزایش دقت و قابلیت تعمیم مدل‌ها استفاده می‌شود. این روش به ویژه در ترکیب با Train/Test Split (تقسیم داده‌ها به مجموعه‌های آموزشی و آزمایشی) به کار می‌رود. در ادامه به توضیح دلایل و مزایای استفاده از Cross-Validation در فرآیند Train/Test Split پرداخته می‌شود:

دلایل و کاربردهای Cross-Validation

1. **ارزیابی دقیق‌تری از عملکرد مدل:**
- با استفاده از Cross-Validation، می‌توانیم به طور دقیق‌تر عملکرد مدل را در برابر داده‌های مختلف ارزیابی کنیم.
- به جای تست کردن مدل تنها روی یک مجموعه آزمایشی (که ممکن است تصادفی انتخاب شده باشد)، با چندین تقسیم‌بندی (folds) می‌توانیم میانگین نتایج را محاسبه کنیم. این به کاهش اثرات تصادفی تقسیم‌بندی کمک می‌کند.

2. **کاهش Overfitting و Underfitting:**
- با ارزیابی مدل روی چندین تنظیم متفاوت از داده‌ها، می‌توانیم از Overfitting جلوگیری کنیم. اگر مدل در هر یک از تقسیم‌ها به خوبی عمل کند، شواهد قوی‌تری برای قابل‌اعتماد بودن آن به دست می‌آوریم.
- Cross-Validation به ما کمک می‌کند تا عملکرد واقعی مدل را در داده‌های جدید پیش‌بینی کنیم و از Underfitting نیز جلوگیری کند.

3. **استفاده بهینه از داده‌ها:**
- در بسیاری از موارد، مجموعه داده‌ها ممکن است کوچک باشد و تقسیم آن به مجموعه‌های آموزشی و آزمایشی ممکن است منجر به از دست رفتن اطلاعات مهم شود.
- با Cross-Validation، از تمام داده‌ها برای آموزش و ارزیابی استفاده می‌شود، زیرا مدل بر روی چندین زیرمجموعه آموزش داده می‌شود و در برابر چندین زیرمجموعه دیگر آزمایش می‌شود.

4. **انتخاب بهتر مدل و تنظیم هایپرپارامترها:**
- Cross-Validation به ما این امکان را می‌دهد که مدل‌های مختلف و هایپرپارامترهای متفاوت را مقایسه کنیم. با ارزیابی مدل‌ها در چندین تقسیم‌بندی، می‌توانیم به انتخاب بهتری برای مدل نهایی برسیم.
- این روش به ما کمک می‌کند تا پارامترهای مدل را بهینه‌سازی کنیم و از انتخاب تصادفی پارامترهایی که ممکن است بر اساس نتیجه‌گیری‌های نادرست باشند، جلوگیری کنیم.

5. **توانایی در تحلیل و تشخیص Bias و Variance:**
- با استفاده از Cross-Validation، می‌توانیم به راحتی متوجه شویم که آیا مدل دچار bias (سوگیری) است یا variance (تنوع) بالایی دارد. این اطلاعات به ما کمک می‌کند تا تصمیمات بهتری برای بهبود مدل بگیریم.

روش‌های مختلف Cross-Validation:

1. **K-fold Cross-Validation:**
- داده‌ها به K بخش تقسیم می‌شوند. هر بار یکی از بخش‌ها به عنوان مجموعه آزمایشی و بقیه به عنوان مجموعه آموزشی استفاده می‌شود.
- این روند K بار تکرار می‌شود و میانگین نتایج جمع‌آوری می‌شود.

2. **Stratified K-fold Cross-Validation:**
- مشابه K-fold است اما برای مسائلی که داده‌ها ممکن است نامتعادل باشند، استفاده می‌شود. در این روش، نسبت کلاس‌ها در هر بخش حفظ می‌شود.

3. **Leave-one-out Cross-Validation (LOOCV):**
- یک نوع خاص از K-fold است که K برابر با تعداد نمونه‌ها است. هر بار تنها یک نمونه به عنوان مجموعه آزمایشی و بقیه به عنوان مجموعه آموزشی استفاده می‌شوند.
- این روش می‌تواند زمان‌بر باشد مگر اینکه داده‌ها بسیار کوچک باشند.

جمع‌بندی:

Cross-Validation ابزاری بسیار قدرتمند و ضروری در فرآیند آموزش و ارزیابی مدل‌های یادگیری ماشین است. با کاهش اثر تصادف در انتخاب مجموعه‌های آزمایشی، افزایش دقت ارزیابی، و بهینه‌سازی انتخاب مدل و پارامترها، Cross-Validation به ما کمک می‌کند تا مدل‌هایی با قابلیت تعمیم بهتر و عملکرد بالاتر بسازیم.

I. Gradient Descent چگونه کار می‌کند؟ ..

Gradient Descent (نزول گرادیان) یکی از الگوریتم‌های بهینه‌سازی پرکاربرد در یادگیری ماشین و یادگیری عمیق است که برای یافتن حداقل یک تابع هزینه (Cost Function) استفاده می‌شود. این الگوریتم به طور تکراری پارامترهای مدل را به‌روزرسانی می‌کند تا تابع هزینه به حداقل برسد. در ادامه به نحوه کار Gradient Descent پرداخته می‌شود:

*1. مفهوم کلی**

هدف Gradient Descent یافتن مقادیر پارامترهای مدل است که تابع هزینه را به حداقل می‌رساند. این کار با محاسبه گرادیان (شیب) تابع هزینه نسبت به پارامترها و به‌روزرسانی پارامترها در جهت مخالف گرادیان انجام می‌شود.

*2. مراحل کار Gradient Descent**

1. **مقداردهی اولیه پارامترها:**
- پارامترهای مدل (مانند وزن‌ها در شبکه‌های عصبی) با مقادیر تصادفی یا صفر مقداردهی می‌شوند.

2. **محاسبه تابع هزینه:**
- تابع هزینه (Cost Function) که نشان‌دهنده خطای مدل است، محاسبه می‌شود. این تابع معمولاً به صورت میانگین مربعات خطا (MSE) یا هر معیار دیگری تعریف می‌شود.

3. **محاسبه گرادیان:**
- گرادیان تابع هزینه نسبت به هر پارامتر محاسبه می‌شود. گرادیان نشان‌دهنده جهت و میزان تغییر تابع هزینه با تغییر پارامترها است.

4. **به‌روزرسانی پارامترها:**
- پارامترها در جهت مخالف گرادیان به‌روزرسانی می‌شوند. این کار با استفاده از فرمول زیر انجام می‌شود:

5. **تکرار مراحل 2 تا 4:**
- مراحل محاسبه تابع هزینه، گرادیان و به‌روزرسانی پارامترها تا زمانی که تابع هزینه به حداقل برسد یا تعداد تکرارها به حداکثر برسد، ادامه می‌یابد.

3. انواع Gradient Descent

1. **Batch Gradient Descent:**
- در هر تکرار، گرادیان بر اساس تمام داده‌های آموزشی محاسبه می‌شود.
- این روش دقیق است اما برای مجموعه‌های داده بزرگ می‌تواند کند باشد.

2. **Stochastic Gradient Descent (SGD):**
- در هر تکرار، گرادیان بر اساس یک نمونه تصادفی از داده‌های آموزشی محاسبه می‌شود.
- این روش سریع‌تر است اما نوسانات بیشتری دارد.

3. **Mini-batch Gradient Descent:**
- در هر تکرار، گرادیان بر اساس یک زیرمجموعه کوچک (mini-batch) از داده‌های آموزشی محاسبه می‌شود.
- این روش تعادلی بین دقت و سرعت ایجاد می‌کند و معمولاً در عمل استفاده می‌شود.

4. نرخ یادگیری (Learning Rate)

- نرخ یادگیری (α) یکی از مهم‌ترین پارامترها در Gradient Descent است. این پارامتر تعیین می‌کند که در هر تکرار، پارامترها چقدر به‌روزرسانی شوند.
- اگر نرخ یادگیری خیلی کوچک باشد، فرآیند بهینه‌سازی کند می‌شود.
- اگر نرخ یادگیری خیلی بزرگ باشد، ممکن است الگوریتم از حداقل تابع هزینه عبور کند و همگرا نشود.

. چالش‌ها و راه‌حل‌ها

1. **محدودیت در همگرایی به حداقل محلی:**
- Gradient Descent ممکن است در حداقل محلی (Local Minimum) گیر کند و نتواند به حداقل سراسری (Global Minimum) برسد.
- راه‌حل: استفاده از تکنیک‌هایی مانند Momentum یا Adam که به فرار از حداقل‌های محلی کمک می‌کنند.

2. **نوسانات در Stochastic Gradient Descent:**
- در SGD، به دلیل استفاده از یک نمونه تصادفی، نوسانات زیادی در به‌روزرسانی پارامترها وجود دارد.
- راه‌حل: استفاده از Mini-batch Gradient Descent یا کاهش تدریجی نرخ یادگیری.

6. جمع‌بندی

Gradient Descent یک الگوریتم بهینه‌سازی قدرتمند است که با محاسبه گرادیان تابع هزینه و به‌روزرسانی پارامترها در جهت مخالف گرادیان، به یافتن حداقل تابع هزینه کمک می‌کند. این الگوریتم در انواع مختلفی مانند Batch, Stochastic و Mini-batch وجود دارد و انتخاب نوع مناسب آن به اندازه مجموعه داده و نیازهای مدل بستگی دارد. نرخ یادگیری نیز یکی از پارامترهای کلیدی است که باید به دقت تنظیم شود تا الگوریتم به درستی همگرا شود.

J. چرا Deep Learning برای پیچیده‌ترین مسائل استفاده می‌شود؟
Deep Learning

، که زیرمجموعه‌ای از یادگیری ماشین است، به دلیل ویژگی‌ها و قابلیت‌های خاصی که دارد، برای حل پیچیده‌ترین مسائل مورد استفاده قرار می‌گیرد. در زیر به چند دلیل اصلی اشاره می‌شود:

1. **مدل‌های عمیق**:
- شبکه‌های عصبی عمیق (Deep Neural Networks) می‌توانند به‌صورت خودکار ویژگی‌های پیچیده‌تری را از داده‌ها استخراج کنند. با لایه‌های متعدد، این شبکه‌ها می‌توانند به طور غیرخطی الگوهای پیچیده‌ای را شناسایی کنند.

2. **عملکرد در داده‌های بزرگ**:
- Deep Learning به‌خوبی بر روی مجموعه داده‌های بزرگ عمل می‌کند. امروزه با افزایش میزان داده‌ها، روش‌های یادگیری عمیق می‌توانند از این داده‌ها برای یادگیری بهتر و دقیق‌تر استفاده کنند.

3. **قابلیت تعمیم بالا**:

- مدل‌های عمیق توانایی بالایی در تعمیم گذاری دارند؛ به این معنی که می‌توانند الگوها را در داده‌های جدید شناسایی کنند که به طور مستقیم بخشی از داده‌های آموزشی نبوده‌اند.

4. **پردازش موازی**:
- ساختار شبکه‌های عصبی به گونه‌ای است که می‌توانند عملیاتی موازی انجام دهند. این ویژگی کمک می‌کند تا زمان پردازش برای آموزش و پیش‌بینی کاهش یابد.

5. **توسعه در حوزه‌های مختلف**:
- Deep Learning در بسیاری از حوزه‌ها از جمله بینایی ماشین (Computer Vision)، پردازش زبان طبیعی (Natural Language Processing)، و سیستم‌های توصیه‌گر به کار رفته است. توانایی انطباق با نیازهای مختلف این حوزه‌ها همچنین باعث گسترش کاربرد این فناوری شده است.

6. **یادگیری بدون نظارت**:
- مدل‌های یادگیری عمیق می‌توانند به‌طور خودکار الگوهای پیچیده را از داده‌های بدون برچسب شناسایی کنند، که این امر کاربرد آن‌ها را در موقعیت‌هایی که داده‌ها کمیاب هستند، افزایش می‌دهد.

به‌طور خلاصه، Deep Learning به دلیل ساختارهای پیچیده، قابلیت پردازش داده‌های بزرگ، و انعطاف‌پذیری در یادگیری از الگوهای جدید، برای حل مسائل پیچیده بسیار مناسب

پایان.....