بنام خدا

مریم رضوانی
شماره دانشجویی: ۹۹۲۱۱۶۰۰۱۹

گزارش پیاده سازی:

مقدمه: این کد پایتون الگوریتم K-Nearest Neighbors را با وزن های یکنواخت برای طبقه بندی مجموعه داده گل زنبق پیاده سازی می کند. همچنین یک فرآیند تنظیم فراپارامتر برای یافتن تعداد بهینه همسایگان انجام می دهد.

ابتدا کتابخانه‌های sklearn.model_selection، و sklearn.datasets برای تقسیم داده‌ها، پیاده‌سازی KNN و بارگذاری مجموعه داده‌های Iris وارد می‌شوند.
تعریف و بارگذاری داده ها: متغیرهای X و y به ترتیب برای ذخیره ویژگی ها و برچسب های هدف مجموعه داده Iris تعریف شده اند. ویژگی های iris.data و iris.target برای بارگذاری داده ها قابل دسترسی هستند. تابع train_test_split داده ها را به مجموعه های آموزشی و آزمایشی با اندازه آزمون 30٪ و حالت تصادفی 8080 برای نتایج قابل تکرار تقسیم می کند.

- **تنظیم فراپارامتر (پیدا کردن k بهینه):**

یک حلقه for از طریق مقادیر مختلف پارامتر n_neighbors، از 3 تا 33 با افزایش 2 تکرار می شود.

- **آموزش و ارزیابی مدل:**

برای هر مقدار n_neighbors، یک طبقه بندی کننده KNN جدید با کلاس KNeighborsClassifier ایجاد می شود.
پارامتر وزن ها روی "uniform" تنظیم شده است، **به این معنی که هر همسایه تأثیر یکسانی در طبقه بندی دارد.**
مدل بر روی داده های آموزشی با استفاده از روش برازش آموزش داده شده است.
عملکرد مدل بر روی داده های آموزشی و آزمایشی با استفاده از روش نمره ارزیابی می شود.
نمرات ارزیابی برای هر دو مجموعه آموزشی و آزمایشی چاپ می شود.

- **انتخاب بهترین مدل:**

کد بهترین نمره آزمون و تعداد همسایه های مربوطه را در حلقه پیگیری می کند. همچنین امتیاز داده آموزشی بهترین مدل را ذخیره می کند.

- **نتایج:**

پس از اتمام حلقه، کد بهترین نمره آزمون، امتیاز داده آموزشی و تعداد بهینه همسایه ها را چاپ می کند.
این کد یک مثال ساده از KNN را با تنظیم هایپرپارامتر نشان می دهد. تنظیم پارامتر n_neighbors برای دستیابی به عملکرد خوب بسیار مهم است. نتایج چاپ شده دیدهایی را در مورد مبادله بین دقت آموزش و تست ارائه می دهد و به تعیین بهترین مدل برای کار خاص کمک می کند.

```
[n_neighbors: 3]        [score train: 0.952381] [score test: 0.977778]
[n_neighbors: 5]        [score train: 0.980952] [score test: 0.955556]
[n_neighbors: 7]        [score train: 0.980952] [score test: 0.955556]
[n_neighbors: 9]        [score train: 0.990476] [score test: 0.955556]
[n_neighbors: 11]       [score train: 0.990476] [score test: 0.977778]
[n_neighbors: 13]       [score train: 0.990476] [score test: 0.977778]
[n_neighbors: 15]       [score train: 0.990476] [score test: 0.977778]
[n_neighbors: 17]       [score train: 0.980952] [score test: 0.977778]
[n_neighbors: 19]       [score train: 0.990476] [score test: 0.933333]
[n_neighbors: 21]       [score train: 0.980952] [score test: 0.911111]
[n_neighbors: 23]       [score train: 0.980952] [score test: 0.933333]
[n_neighbors: 25]       [score train: 0.961905] [score test: 0.933333]
[n_neighbors: 27]       [score train: 0.971429] [score test: 0.933333]
[n_neighbors: 29]       [score train: 0.980952] [score test: 0.911111]
[n_neighbors: 31]       [score train: 0.971429] [score test: 0.911111]

best test score is [0.9778] & train score is [0.9905] for neighbors [11]
```

از تابع confusion_matrix برای تجزیه و تحلیل عملکرد یک مدل KNN قبلا آموزش دیده برای مجموعه داده های آموزشی و آزمایشی استفاده می‌کنیم.

توضیح الگوریتم KNN با وزن‌های مبتنی بر فاصله:

مقدمه: این کد پایتون الگوریتم K-Nearest Neighbors (KNN) را با وزن های مبتنی بر فاصله برای طبقه بندی مجموعه داده گل زنبق پیاده سازی می کند. همچنین یک فرآیند تنظیم فراپارامتر برای یافتن تعداد بهینه همسایگان انجام می دهد.

مشابه کد قبلی، کتابخانه‌های مورد نیاز را importمی‌کنیم.

همان فرآیند قبلی برای تعریف و بارگذاری مجموعه داده های Iris در متغیرهای X و y و تقسیم آن به مجموعه های آموزشی و آزمایشی با نسبت 70/30 و حالت تصادفی 8080 استفاده می شود.

- **تنظیم فراپارامتر:** کد از طریق مقادیر مختلف پارامتر n_neighbors، از 3 تا 33 با افزایش 2 تکرار می شود.
- **آموزش و ارزیابی مدل:**

در حلقه، یک طبقه بندی کننده KNN جدید برای هر مقدار n_neighbors ایجاد می شود. **با این حال، این بار پارامتر وزن ها روی "فاصله" تنظیم می شود که وزن های بالاتر را به همسایگان نزدیک تر و وزن های پایین تر را به همسایگان دورتر اختصاص می دهد.** سپس مدل با استفاده از روش‌های قبلی آموزش داده و بر روی داده‌های آموزش و آزمایش ارزیابی می‌شود.

نتایج شامل تعداد همسایگان، امتیاز آموزش و امتیاز آزمون چاپ می شود.

- **انتخاب بهترین مدل:**

این کد بهترین نمره آزمون، امتیاز داده آموزشی و تعداد همسایگان مربوطه را در طول حلقه ردیابی می کند.

- **نتایج:**

پس از تکرار همه مقادیر، کد بهترین نمره آزمون، امتیاز داده آموزشی و تعداد بهینه همسایه ها را چاپ می کند.

- **تفاوت های اصلی این بخش با با وزن های یکنواخت knn**:

این کد از وزن‌های مبتنی بر فاصله استفاده می‌کند که تأثیر همسایگان نزدیک‌تر را در فرآیند طبقه‌بندی اولویت می‌دهد. این به طور بالقوه می تواند منجر به عملکرد بهتر مجموعه داده با توزیع کلاس غیر یکنواخت شود. شباهت با وزن های یکنواخت: فرآیند تنظیم فراپارامتر و ارزیابی مدل هنوز با استفاده از تکنیک‌های مشابه انجام می‌شود.

```
[n_neighbors: 3]        [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 5]        [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 7]        [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 9]        [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 11]       [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 13]       [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 15]       [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 17]       [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 19]       [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 21]       [score train: 1.000000] [score test: 0.977778]
[n_neighbors: 23]       [score train: 1.000000] [score test: 0.955556]
[n_neighbors: 25]       [score train: 1.000000] [score test: 0.933333]
[n_neighbors: 27]       [score train: 1.000000] [score test: 0.933333]
[n_neighbors: 29]       [score train: 1.000000] [score test: 0.933333]
[n_neighbors: 31]       [score train: 1.000000] [score test: 0.933333]

best test score is [0.9778] & train score is [1.0000] for neighbors [3]
```

و در نهایت رسم مرز تصمیم برای دو حالت:

تابع load_iris دوباره فراخوانی می شود، این بار as_frame=True را تنظیم می کنیم تا داده ها را به عنوان یک Pandas DataFrame برای دستکاری آسان تر نشان داده شود. سپس تنها ویژگی های "sepal length (cm)" و "sepal width (cm)" را از iris.data DataFrame انتخاب می‌کنیم و آن‌ها را به متغیر X اختصاص می‌دهیم . متغیر y همچنان برچسب های هدف را نگه می دارد.

داده ها دوباره با استفاده از train_test_split با همان پارامترها به مجموعه های آموزشی و آزمایشی تقسیم می شوند.

تجسم مرز تصمیم با وزن های مختلف: یک حلقه از طریق دو گزینه وزنی تکرار می شود: "uniform" و "distance".

برای هر طرح وزنی: یک طبقه‌بندی‌کننده KNN با بهترین تعداد همسایه‌ها (best_neighbors) که قبلاً تعیین شده بود آموزش داده می‌شود.

کلاس DecisionBoundaryDisplay از scikit-learn برای تجسم مرزهای تصمیم استفاده می‌شود

متد from_estimator فراخوانی می‌شود که مدل آموزش دیده، داده‌های آزمایشی و ویژگی‌های نمودار مورد نظر را ارسال می‌کند.

روش pcolormesh یک مش پر رنگ ایجاد می کند که احتمالات کلاس پیش بینی شده را برای نقاط داده مختلف نشان می دهد. تابع scatter نقاط داده را با رنگ های مختلف بر اساس کلاس های واقعی آنها ترسیم می کند.

سه روش یادگیری ماشین با استفاده از RBF:

1 :RBF. ماشین بردار پشتیبان تابع پایه شعاعی (RBF) (SVM): این روشی است که به طور گسترده برای کارهای طبقه بندی استفاده می شود، به ویژه هنگامی که با داده های غیر خطی سروکار داریم. RBF SVM داده های ورودی را به

یک فضای ویژگی با ابعاد بالاتر نگاشت می کند که در آن یک ابر صفحه خطی می تواند کلاس ها را از هم جدا کند. تابع هسته مورد استفاده در RBF SVM یک تابع پایه شعاعی است که شباهت بین نقاط داده را بر اساس فاصله اقلیدسی آنها اندازه گیری می کند.

مزایا: داده های غیر خطی را به خوبی مدیریت می کند. چند فراپارامتر برای تنظیم. معایب: می تواند از نظر محاسباتی برای مجموعه داده های بزرگ گران باشد. حساس به انتخاب پارامتر هسته (گاما).

2. شبکه عصبی تابع پایه شعاعی (RBFNN): این نوع شبکه عصبی از RBF به عنوان توابع فعال سازی در لایه پنهان استفاده می کند. RBFNN ها تقریبگرهای جهانی هستند، به این معنی که می توانند هر تابع پیوسته را با دقت دلخواه تقریب بزنند. مانند RBF SVM، این روش نیز در وظایف رگرسیون غیر خطی برتری دارد.

مزایا: می تواند روابط پیچیده بین متغیرهای ورودی و خروجی را بیاموزد. آموزش نسبتا سریع برای مجموعه داده های کوچک.

معایب: اگر منظم نشود، می تواند مستعد بیش از حد برازش شود. به انتخاب دقیق تعداد واحدهای پنهان و پارامترهای RBF نیاز دارد.

3. رگرسیون فرآیند گاوسی (GPR): این یک رویکرد بیزی ناپارامتریک برای وظایف رگرسیونی است. GPR قبل از مدل سازی رابطه بین متغیرهای ورودی و خروجی از یک فرآیند گاوسی استفاده می کند. تابع هسته در GPR معمولاً یک تابع پایه شعاعی است که باور قبلی در مورد اینکه چگونه تابع با داده های ورودی متفاوت است را رمزگذاری می کند.

مزایا: تخمین هایی از عدم قطعیت همراه با پیش بینی ها را ارائه می دهد. می تواند داده های نویز را به خوبی مدیریت کند.

معایب: می تواند از نظر محاسباتی برای مجموعه داده های بزرگ گران باشد. به انتخاب دقیق پارامترهای هسته نیاز دارد.

اینها فقط سه نمونه از روشهای یادگیری ماشینی هستند که از RBFها استفاده می کنند. روش های قابل توجه دیگر عبارتند از درون یابی RBF، خوشه بندی RBF و تشخیص ناهنجاری RBF. هر روش نقاط قوت و ضعف خاص خود را دارد و بهترین انتخاب برای یک مسئله به عوامل مختلفی مانند ماهیت داده ها، نوع کار و منابع محاسباتی بستگی دارد.

4. خوشه بندی RBF: خوشه بندی RBF (تابع پایه شعاعی) نوعی الگوریتم یادگیری بدون نظارت است که برای گروه بندی نقاط داده مشابه در خوشه ها استفاده می شود. از مفهوم توابع پایه شعاعی (RBFs) برای اندازه گیری شباهت بین نقاط داده و تعیین عضویت خوشه آن ها استفاده می کند. در اینجا یک تفکیک از روند است:

1. آماده سازی داده ها: داده ها از قبل پردازش شده و در صورت لزوم مقیاس بندی می شوند. تعداد خوشه های مورد نظر (K) از قبل تعیین می شود.
2. راه اندازی مرکز خوشه: K نقاط داده به طور تصادفی به عنوان مراکز خوشه اولیه انتخاب می شوند. روش دیگر، الگوریتم های خوشه بندی مانند K-means را می توان برای مقداردهی اولیه مراکز استفاده کرد.
3. محاسبه فعال سازی RBF: برای هر نقطه داده: فاصله بین نقطه داده و هر مرکز خوشه را محاسبه کنید. از یک تابع هسته RBF برای تبدیل فاصله به امتیاز شباهت استفاده کنید.
4. انتساب خوشه: هر نقطه داده را به خوشه ای با بیشترین امتیاز شباهت اختصاص دهید.
5. به روز رسانی مرکز خوشه: مراکز خوشه را به میانگین تمام نقاط داده اختصاص داده شده به هر خوشه به روز کنید.
6. مراحل 3-5 را تکرار کنید:

مراحل 3-5 را تکرار کنید تا زمانی که مراکز خوشه تثبیت شوند یا یک معیار توقف از پیش تعریف شده برآورده شود.

هسته های RBF:

هسته های RBF نقش مهمی در خوشه بندی RBF دارند. آنها توابعی هستند که فاصله اقلیدسی بین نقاط داده را به یک امتیاز شباهت ترسیم می کنند. هسته های رایج RBF عبارتند از:

هسته گاوسی: این هسته وزن‌های بالاتری را به نقاط داده نزدیک‌تر اختصاص می‌دهد و در نتیجه مرزهای خوشه صاف‌تر می‌شود.

هسته چند چهارگانه: این هسته وزن های بالاتری را به نقاط داده بسیار نزدیک و بسیار دور اختصاص می دهد و به طور بالقوه خوشه های متنوع تری را شناسایی می کند.
مزایای خوشه بندی RBF: داده های غیر خطی را به خوبی مدیریت می کند. مقاوم در برابر داده های پرت و پر سر و صدا. نسبتا سریع در مقایسه با برخی دیگر از الگوریتم های خوشه بندی.
معایب خوشه بندی RBF: به انتخاب تابع هسته و پارامترهای آن حساس است. تعداد بهینه خوشه ها باید از قبل تعیین شود. کاربردهای خوشه بندی RBF: تقسیم بندی مشتریان تقسیم بندی تصویر تشخیص ناهنجاری تجزیه و تحلیل بیان ژن به طور کلی، خوشه بندی RBF یک ابزار قدرتمند و همه کاره برای گروه بندی نقاط داده در خوشه های معنی دار است. با این حال، برای دستیابی به نتایج بهینه، مهم است که انتخاب تابع هسته و پارامترهای آن و همچنین تعداد خوشه های مورد نظر را به دقت در نظر بگیرید.

توضیح کد خوشه بندی RBF:

این قطعه کد، خوشه بندی RBF را با استفاده از MiniBatchKMeans با هسته گاوسی پیاده سازی می کند و عملکرد آن را با KMeans معمولی با استفاده از امتیاز silhouette به عنوان معیار ارزیابی مقایسه می کند. این کد همچنین نتایج خوشه بندی را با استفاده از PCA برای درک بهتر ساختار خوشه به تصویر می کشد.
این کد پایتون خوشه‌بندی RBF را برای مجموعه داده Iris با استفاده از کتابخانه‌های scikit-learn پیاده‌سازی می‌کند و نمودار آن را رسم می‌کند.
کتابخانه‌هایی مانند پانداها، scikit-learn و matplotlib وارد می‌شوند.
بارگیری مجموعه داده ایریس: مجموعه داده Iris با استفاده از load_iris از scikit-learn بارگیری می شود.
استاندارد کردن داده ها: داده ها با استفاده از StandardScaler برای عملکرد بهتر خوشه بندی استاندارد شده اند.
تعداد خوشه ها: تعداد خوشه های مورد نظر 3 عدد تنظیم شده است که مربوط به سه گونه گل زنبق است.
خوشه بندی RBF با MiniBatchK به این معنی است: یک مدل خوشه‌بندی RBF با استفاده از MiniBatchKMeans ایجاد می‌شود.
این پیاده سازی از یک رویکرد مینی دسته ای برای آموزش سریع‌تر استفاده می کند و از محدودیت های حافظه جلوگیری می کند. تابع هسته روی **هسته گاوسی** پیش فرض تنظیم شده است. سایر پارامترها مانند init، max_iter و random_state برای همگرایی و تکرارپذیری بهتر تنظیم می شوند.

ارزیابی عملکرد: امتیاز silhouette با استفاده از silhouette_score برای ارزیابی عملکرد خوشه‌بندی محاسبه می‌شود. این امتیاز نشان می دهد که نقاط داده بر اساس فاصله آن‌ها تا خوشه های اختصاص داده شده و خوشه های همسایه چقدر خوب خوشه بندی شده اند.
مقایسه منظم KMeans: خوشه بندی منظم KMeans برای مقایسه با استفاده از KMeans انجام می شود. برای این مدل همان امتیاز silhouette محاسبه شده است.
تجسم: PCA برای کاهش ابعاد داده برای تجسم استفاده می شود.
نمودارهای پراکندگی برای تجسم خوشه های تشکیل شده توسط هر دو مدل RBF و KMeans ایجاد می شوند.
خلاصه گزارش: این کد کاربرد خوشه‌بندی RBF با MiniBatchKMeans را برای مجموعه داده Iris نشان می‌دهد. پارامترهای انتخاب شده و تکنیک های تجسم، درک روشنی از نحوه خوشه بندی داده ها و عملکرد الگوریتم انتخاب شده ارائه می دهند.
مشاهدات: امتیاز silhouette برای خوشه‌بندی RBF بالاتر از KMeans است، که نشان می‌دهد خوشه‌بندی RBF ممکن است مناسب‌تر برای این مجموعه داده باشد.

تجسم‌ها خوشه‌های متمایز تشکیل‌شده توسط هر دو مدل را نشان می‌دهند، با خوشه‌های RBF کمی فشرده‌تر و به خوبی تعریف شده‌تر به نظر می‌رسند.
ملاحظات بیشتر: بررسی توابع مختلف هسته و فراپارامترها برای خوشه‌بندی RBF می‌تواند نتایج را به طور بالقوه بهبود بخشد. اجرای معیارهای ارزیابی اضافی مانند امتیاز Calinski-Harabasz یا شاخص Davies-Bouldin می تواند بینش بیشتری در مورد کیفیت خوشه ارائه دهد.