دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

گزارش هفتم

مقایسه روش های مختلف طبقه بندی روی یک مساله benchmark

نگارش

سروش آریانا

استاد

دکتر مهدی قطعی

خرداد ۱۴۰۰

مقدمه

الگوریتم‌های با ناظر (Supervised) الگوریتم‌هایی هستند که از داده دارای برچسب (Labeled) برای یادگیری استفاده می‌کنند. الگوریتم پس از درک کردن داده، با تطبیق الگوها با داده‌های بدون برچسب، تعیین می‌کند که چه برچسبی باید به داده‌های جدید داده شود.

الگوریتم‌های با ناظر به دو دسته تقسیم می‌شوند:

- طبقه‌بندی (Classification)
- رگرسیون (Regression)

طبقه‌بندی، مساله‌ای است که در آن الگوریتم تشخیص می‌دهد یکسری مشاهدات به کدامیک از دسته‌ها (زیر جمعیت‌ها) تعلق دارد. برای مثال اختصاص دادن ایمیل‌ها به دسته‌های هرزنامه (Spam) و غیرهرزنامه (Non-Spam) نمونه‌ای از یک مساله یک مساله طبقه‌بندی است.

در این گزارش به بررسی دو نمونه الگوریتم طبقه‌بندی بر روی داده مجموعه ارقام دست‌نویس فارسی می‌پردازیم. کد های مربوطه در کولب قابل دریافت و مشاهده است که در کنار فایل pdf گزارش با نام AI_Assignment_7.ipynb ارسال شده است.

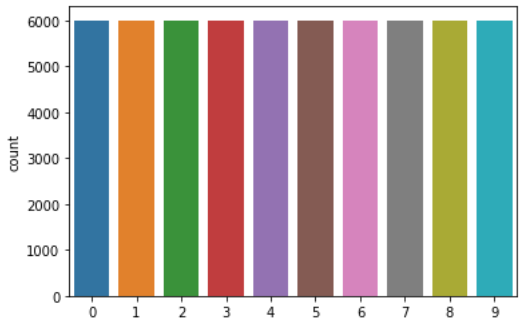
۱- دیتاست

دیتای ارقام دست‌نویس یا Mnist یکی از دیتاهای محبوب برای بررسی الگوریتم‌های ماشین لرنینگ، مخصوصا الگوریتم‌های طبقه‌بندی، است. در این پروژه از دیتاست هدی که یک مجموعه ارقام دست‌نویس فارسی است استفاده کردیم. این دیتاست شامل ۱۰۲۳۵۳ نمونه دست نوشته سیاه و سفید است.

این مجموعه اولین مجموعه‌ی بزرگ ارقام دست‌نویس فارسی است که از حدود ۱۲۰۰۰ فرم ثبت‌نام آزمون سراسری کارشناسی ارشد ۱۳۸۴ و آزمون کاردانی پیوسته‌ی دانشگاه جامع علمی کاربردی سال ۱۳۸۳ استخراج شده است.

رقم ۰	رقم ۱	رقم ۲	رقم ۳	رقم ٤	رقم ٥	رقم ٦	رقم ٧	رقم ٨	رقم ٩
10070	10330	9923	10334	10333	10110	10254	10363	10264	10371

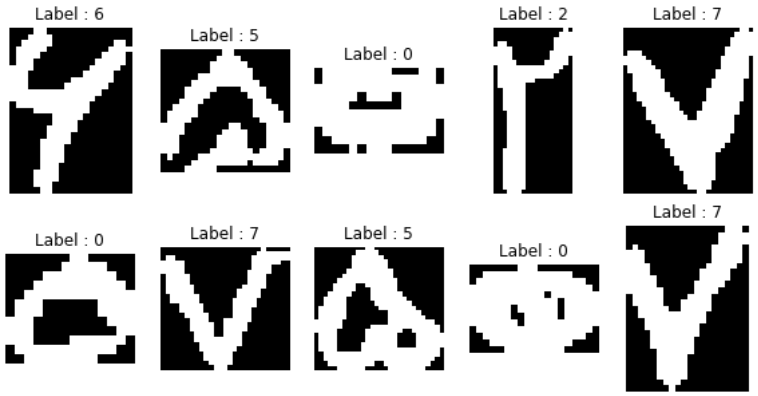
شکل ۱ توزیع کلاس‌های مختلف داده



شکل۲ نمودار توزیع کلاس‌های مختلف در دیتای ترین

همان‌طور که در شکل ۱ و شکل۲ مشخص است توزیع کلاس‌های مختلف در این دیتاست تا حد بسیار زیادی یکسان است و هر کلاس‌ها در دیتای ترین حدودا ۶۰۰۰ نمونه و در دیتای تست حدودا ۲۰۰۰ نمونه دارد.

این دیتاست از سایت farsiocr قابل دریافت برای استفاده تحقیقاتی است.



شکل۳ داده‌های خام دیتاست

۲-۱- پیش‌پردازش

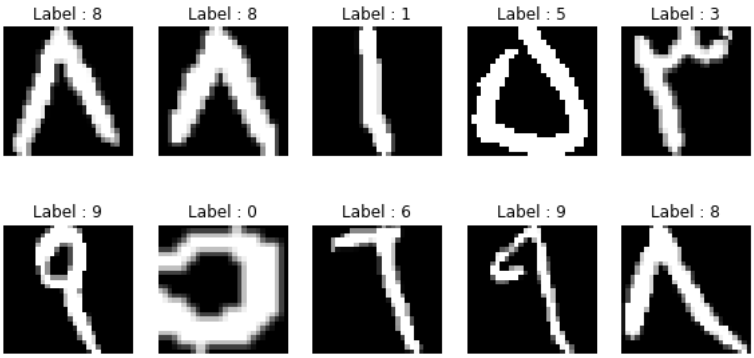
همانطور که در شکل۳ پیداست، در اولین نگاه به دیتاست پس از خواندن داده‌ها اولین مشکلی که به نظر می‌رسد یکسان نبودن ابعاد تصاویر است. برای حل این مشکل، ابتدا تصاویر را برای رسیدن به طول و عرض یکسان لایه‌گذاری (padding) کردیم و سپس آن‌ها را به سایز دلخواه تغییر اندازه دادیم.

```
def reshape_images(dataset, size):
  result = []
  for d in dataset:
    height, width = d.shape
    ww = max(height, width)
    pad = np.zeros((ww, ww))
    xx = (ww - width) // 2
    yy = (ww - height) // 2
    pad[yy:yy+height, xx:xx+width] = d
    result.append(cv2.resize(pad, (size, size)))
  return np.array(result)
```

کد ۱ تابع مربوط به پیش‌پردازش تصاویر

در نهایت، همانطور که در شکل۴ مشخص است، تمامی تصاویر به یک اندازه تبدیل شده‌اند و تنها کاری که برای آماده سازی جهت استفاده در الگوریتم‌های طبقه‌بندی لازم است انجام بدهیم تغییر شکل این تصاویر از حالت دو بعدی به حالت تک بعدی است.

برای مثال اگر سایز تصاویر ما ۲۸ در ۲۸ باشد، پس از تغییر شکل تصاویر به حالت تک بعدی، هر تصویر یک آرایه ۷۸۴تایی خواهد شد.



شکل۴ تصاویر دیتاست پس از انجام عملیات پیش‌پردازش

۲- الگوریتم‌های طبقه بندی

برای این گزارش ما دو الگوریتم درخت تصمیم (Decision Tree) و جنگل تصادفی (Random Forest) و پیاده‌سازی‌های آنها در کتابخانه ScikitLearn پایتون استفاده کردیم که در ادامه به توضیح مختصر هر الگوریتم می‌پردازیم.

۲-۲- درخت تصمیم

هدف از استفاده درخت تصمیم ایجاد یک مدل آموزشی است که می‌تواند با یادگیری قوانین ساده تصمیم‌گیری استنباط شده از داده‌های آموزش، در مرحله استنباط از این قوانین برای پیش‌بینی کلاس استفاده کند.

در درخت تصمیم، برای پیشبینی برچسب کلاس برای هر رکورد، ما از ریشه درخت شروع می‌کنیم و بر اساس مقایسه شاخه به آن مقدار را دنبال می‌کنیم و به گره بعدی می‌رویم.

۲-۳- جنگل تصادفی

در روش جنگل تصادفی، بصورت خلاصه، ما چندین درخت تصمیم ساخته و آن‌ها را با هم ادغام می‌کند تا به یک پیشبینی دقیق و پایدارتر برسد. روش جنگل تصادفی یکی از الگوریتم‌های اسان و منعطف است که در بیشتر مواقع حتی بدون تنظیم هایپرپارامترها نتیجه بسیار خوبی بدست می‌آورد.

۳- توضیح کد

برای خواندن دیتا از فرمت cdb به آرایه پایتون، از کلاس پایتون معرفی شده در سایت رسمی هدی استفاده کردیم. فانکشن read_hoda_cdb به عنوان ورودی آدرس فایل cdb حاوی دیتای ارقام را دریافت کرده و به عنوان خروجی، داده را به عنوان آرایه پایتون تحویل می‌دهد.

برای به تصویر کشیدن این آرایه بصورت تصویر از کتابخانه matlibplot استفاده کردیم. به اینصورت که ابتدا، به تعداد دلخواه، تعدادی از تصاویر را از داده انتخاب کرده آرایه مربوط به آن تصویر و برچسب آن تصویر را دریافت کرده و با فانکشن imshow در کتابخانه آن‌ها را به تصویر کشیدیم.

همانطور که اشاره شد در این گزارش از پیاده‌سازی‌های این الگوریتم‌ها در کتابخانه ScikitLearn استفاده کردیم. استفاده از این پیاده‌سازی‌ها به اینصورت است که ابتدا یک آبجکت از کلاس الگوریتم دلخواه (DecisionTreeClassifier یا RandomForestClassifier) ایجاد کرده و سپس فانکشن fit را که به عنوان ورودی آرایه‌ای از داده آموزش و برچسب-های ان را دریافت می‌کند صدا می‌زنیم. سپس برای استفاده از مدل برای پیشبینی داده تست از متد predict استفاده می-کنیم که به عنوان ورودی آرایه دادگان تست را دریافت می‌کند. یکی از پارامترهای مهم در پیاده سازی RandomForestClassifier، تعداد درخت در کل جنگل است که با پارامتر n_estimators تنظیم می‌شود که به صورت پیشفرض این عدد روی ۱۰۰ تنظیم شده است.

برای محاسبه دقت مدل از معیار دقت استفاده کردیم که با عنوان accuracy_score در کتابخانه ScikitLearn پیاده-سازی شده است و به عنوان ورودی آرایه‌ای از برچسب‌های واقعی و برچسب‌های پیشبینی شده (خروجی متد predict) را دریافت کرده و دقت مدل را به عنوان خروجی برمی‌گرداند.

برای ساخت ماتریس درهم‌ریختگی از پیاده‌سازی آن در کتابخانه به عنوان plot_confusion_matrix استفاده کردیم. این فانکشن به عنوان ورودی مدل مورد نظر، داده تست و برچسب‌های ورودی آن را دریافت می‌کند و پس از استفاده از مدل برای پیشبینی برچسب‌ها روی داده تست که به عنوان ورودی داده شده، نمودار را به تصویر می‌کشد.
برای کشیدن نمودار میله‌ای از فانکشن countplot در کتابخانه seaborn استفاده کردیم که به عنوان ورودی یک آرایه از اعداد (در استفاده ما برچسب‌های داده) دریافت کرده و نمودار میله‌ای مربوط به آن را به تصویر می‌کشد.

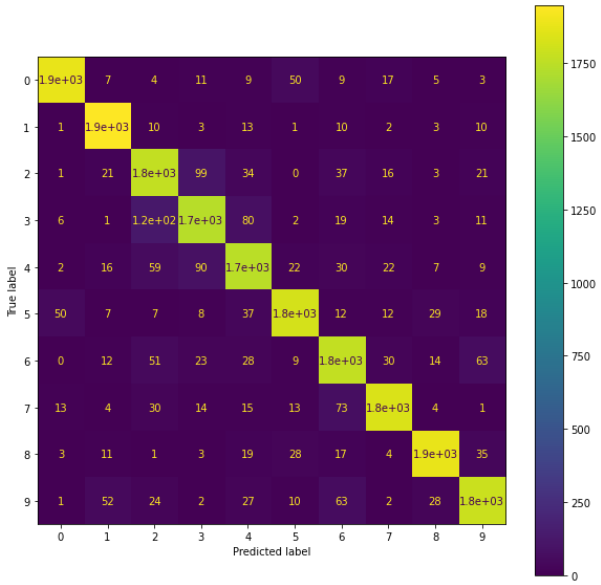
۴- بررسی و آنالیز

معیار اصلی‌ای که برای بررسی نتایج این دو الگوریتم استفاده کردیم، دقت (Accuracy) است که تعریف آن بصورت ساده نسبت تعداد پیش‌بینی‌های درست مدل نسبت به تعداد کل پیشبینی‌های مدل است. همچنین برای تصویری کردن خروجی

مدل، نمودار ماتریس درهم‌ریختگی (Confusion Matrix) که نمایانگری ساده از کلاس‌های پیشبینی شده توسط مدل و برچسب اصلی داده است را به تصویر کشیدیم.

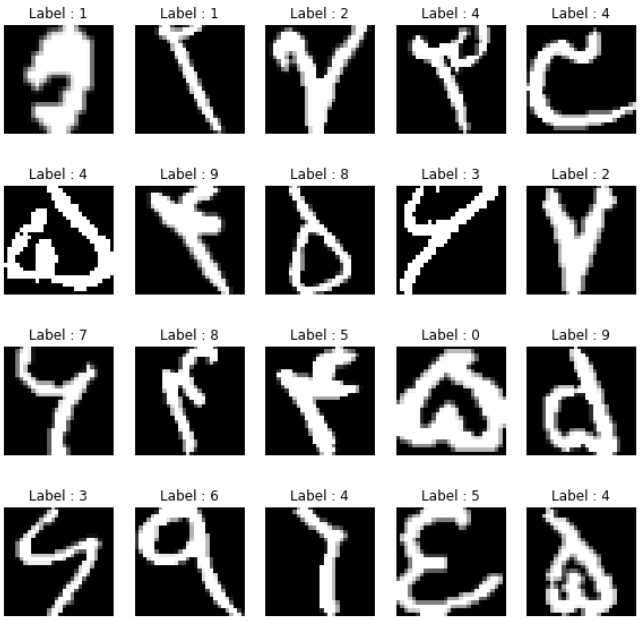
```
0.9083
time: 19.5 s (started: 2021-05-26 18:53:52 +00:00)
```

شکل ۵ دقت و زمان اجرای الگوریتم طبقه‌بندی درخت تصمیم



شکل ۶ ماتریس درهم‌ریختگی الگوریتم درخت تصمیم

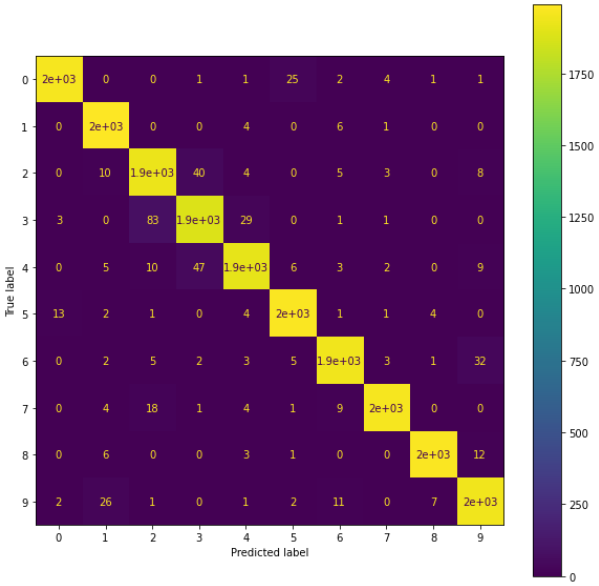
همانطور که در شکل ۵ مشخص است دقت الگوریتم درخت تصمیم روی دادگان تست حدودا برابر با ۹۰ درصد است که دقت نسبتا خوبی است و مرحله آموزش این الگوریتم حدودا ۲۰ ثانیه به طول انجامیده است. علاوه بر این در شکل ۶ می‌توان مشاهده کرد که بیشترین خطا، پیشبینی عدد ۲ به جای ۳ بوده که در ۱۲۰۰ نمونه این خطا انجام شده است و کم‌خطاترین کلاس پیشبینی شده، متعلق به شماره ۱ بوده است.



شکل ۷ نمونه خطاهای الگوریتم درخت تصمیم – برچسب‌ها پیشبینی الگوریتم است

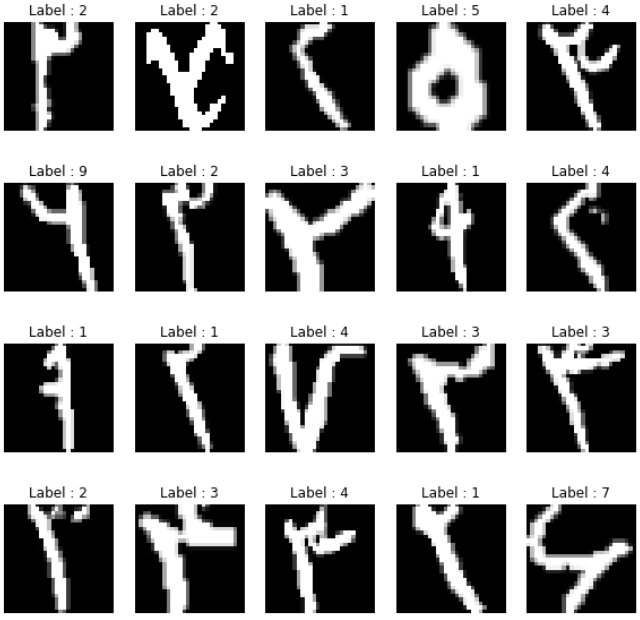
```
0.97485
time: 40.3 s (started: 2021-05-27 07:57:04 +00:00)
```

شکل ۸ دقت و زمان اجرای الگوریتم جنگل تصادفی



شکل ۹ ماتریس درهم‌ریختگی الگوریتم جنگل تصادفی

با توجه به خروجی الگوریتم در شکل ۸، می‌توان مشاهده کرد که الگوریتم به دقت بسیار خوب ۹۷ درصد رسیده است. همچنین با توجه به ماتریس درهم‌ریختگی شکل ۹ می‌توان مشاهده کرد که اکثر خطاها تک رقمی هستند و همچنان پیشبینی ۲ به جای ۳ بیشترین تعداد خطا را در کل داده تست دارد. همچنین در **شکل ۱۰** قابل مشاهده است که خطاهای الگوریتم جنگل تصادفی بسیار پیچیده‌تر از خطاهایی است که توسط الگوریتم درخت تصمیم ایجاد شده است.



شکل ۱۰ نمونه خطاهای الگوریتم جنگل تصادفی – برچسب‌ها پیشبینی الگوریتم است

۵- نتیجه گیری

همانطور که مشاهده شد، هر دو الگوریتم به دقت بسیار خوبی بر روی داده ارقام دست‌نوشته فارسی دست پیدا کردند اما الگوریتم جنگل تصادفی دقت بهتری نسبت به الگوریتم درخت تصمیم دارد و آن از پیچیدگی بیشتر آن الگوریتم ناشی می-شود. همانطور که اشاره شد الگوریتم جنگل تصادفی حتی بدون تنظیم هایپر پارامترها قادر به رسیدن به دقت خیلی خوب است، همانطور که در داده ما نیز همین نتیجه را داد. اما از طرفی آموزش جنگل تصادفی بر روی داده ساده ما نزدیک به ۲ برابر بیشتر نسبت به آموزش درخت تصمیم ساده زمان گرفته است که این مورد را در هنگام انتخاب این الگوریتم باید در نظر گرفت.

منابع و مراجع

[1] "Plotting with matplotlib," [Online]. Available: https://pandas.pydata.org/pandas-docs/version/0.13/visualization.html.

[2] "Random Forest," [Online]. Available: https://en.wikipedia.org/wiki/Random_forest.

[3] "Decision tree learning," [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree_learning.

[4] "مجموعه ارقام دستنویس هدی," [Online]. Available: http://farsiocr.ir/%D9%85%D8%AC%D9%85%D9%88%D8%B9%D9%87-%D8%AF%D8%A7%D8%AF%D9%87/%D9%85%D8%AC%D9%85%D9%88%D8%B9%D9%87-%D8%A7%D8%B1%D9%82%D8%A7%D9%85-%D8%AF%D8%B3%D8%AA%D9%86%D9%88%DB%8C%D8%B3-%D9%87%D8%AF%DB%8C/.

[5] "Plot Confusion Matrix," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.plot_confusion_matrix.html.