

به نام خدا



دانشکده مهندسی کامپیوتر

دانشگاه صنعتی امیرکبیر  
دانشکده مهندسی کامپیوتر

استاد درس: دکتر امین ناظرفرد

پاییز ۱۳۹۹

درس یادگیری ماشین

سری ۴ تمرین‌ها

سروش مهدی  
شماره دانشجویی: ۹۹۱۳۱۰۵۰



فهرست مطالب

		سوالات تشریحی	۱
۱	.....	۱.۱	۱
۱	.....	۱.۱.۱	۱
۱	.....	۲.۱.۱	۱
۱	.....	۳.۱.۱	۱
۱	.....	۴.۱.۱	۱
۱	.....	۵.۱.۱	۱
۱	.....	۶.۱.۱	۱
۱	.....	۲.۱	۱
۱	.....	۳.۱	۱
۲	.....	۴.۱	۱
۳		سوالات پیاده سازی	۲
۳	.....	SVM	۱.۲
۳	.....	۱.۱.۲	۱
۶	.....	۲.۱.۲	۱
۶	.....	۳.۱.۲	۱
۶	.....	۱.۲.۲	۱
۹	.....	۲.۲.۲	۱
۹	.....	۳.۲.۲	۱

## ۱ سوالات تشریحی

۱.۱

۱.۱.۱

این مورد اشتباه است زیرا در این الگوریتم تعداد پارامتر هایی که مدل میخواهد یاد بگیرد بستگی به تعداد داده ها دارد و باید ضرایب الفا را پیدا کند و تعداد این ضرایب با بالا رفتن تعداد داده ها بیشتر میشود

۲.۱.۱

خیر ممکن است یک مجموعه داده خطی جدا پذیر با یک کرنل گاوی اموزش داده شود و به مقدار حاشیه بیشتری برسد نسبت به مدلی که با کرنل خطی اموزش داده شده اما روی داده اموزش دچار بیش برآش شود در نتیجه روی داده های تست عملکرد خوبی نداشته باشد.

۳.۱.۱

خیر اگر کرنل بیش از حد پیچیده در نظر بگیریم و پارامتر رگولاریزیشن را مناسب انتخاب نکنیم بیش برآش رخ خواهد داد مثال قسمت قبل را در نظر بگیرید

۴.۱.۱

خیر داده های نویز ممکن است باعث کاهش دقت این مدل ها شوند که با استفاده از پارامتر رگولاریزیشن میتوان تا حدودی ان را کنترل کرد.

۵.۱.۱

اگر هر دسته بندی کننده فقط کمی بهتر از دسته بند تصادفی عمل کند این عبارت درست است دز غیر این صورت خیر.

۶.۱.۱

اگر یک دسته بندی کننده عملکردی بدتر از رندوم داشته باشد وزن آن منفی میشود. پس این عبارت غلط است.

۲.۱

شکل سمت راست مربوط به مقدار ده شکل وسط مربوط به مقدار دو دهم و شکل سمت چپ مربوط به مقدار یک میباشد تاثیر این پارامتر به این صورت است که کاهش ان محدوده تاثیر هر نمونه داده بیشتر میشود پس هر چقدر بیشتر باشد مرز های تصمیم انعطاف پذیرتر میشوند و اگر کمتر باشد مرز تصمیم خطی تر میشود.

۳.۱

در روش hard voting ما از هر دسته بندی کننده کلاسی را که بیشترین احتمال را دارد انتخاب میکنیم و سپس بین همه این کلاس ها کلاسی که بیشترین رای را دارد به عنوان خروجی انتخاب میکنیم. اما در روش voting hard ما ابتدا برای هر دسته بندی کننده احتمالی که برای هر کلاس پیش بینی میکند را بدست می اوریم سپس برای هر کلاس میانگین این احتمالات را حساب میکنیم و هر کلاس که مقدار بیشتری داشته باشد را به عنوان کلاس نهایی انتخاب میکنیم.



دانشگاه صنعتی کالجیور

## درس یادگیری ماشین

### ۴.۱

در حالت voting hard رای دسته بندی کننده اول کلاس ۲ میباشد و رای دسته بندی کننده های دیگر کلاس یک میباشد که با توجه به وزن دسته بندی کننده ها کلاس ۲ دو رای و کلاس ۱ سه رای می اورد و در نهایت کلاس یک انتخاب میشود در حالت voting soft باید میانگین وزن دار احتمال ها محاسبه شود. این مقدار برای کلاس یک برابر سی و دو صدم میباشد. برای کلاس دوم برابر سی و هشت صدم میباشد. برای کلاس سوم برابر سه دهم میباشد. پس در این حالت کلاس دوم انتخاب میشود.

## ۲ سوالات پیاده سازی

در این بخش برای تمامی مدل ها از کتابخانه `sklearn` استفاده شده است.

### SVM ۱.۲

#### ۱.۱.۲

در این قسمت ما ابتدا مدل ها با کرنل ها و پارامتر های مختلف را اماده سازی میکنیم.تابع `model` به عنوان ورودی تابع مدل که در اینجا یکی از مدل های کتابخانه SKlearn هست و یک دیکشنری که کلید های ان عناوین پارامتر ها و مقادیر ان لیست هایی از مقادیر مختلف پارامتر ها میباشد را میگیرد. بعد از این تابع ما یک دیکشنری داریم که کلید های ان اسم مدل ها به همراه پارامتر های تعیین کننده هست و مقادیر ان مدل ها میباشند. که این دیکشنری بوسیله تابع بالا محاسبه میشود. کد های این قسمت ها در صفحات بعدی امده اند.

ما در ابتدای پیش و پنج درصد داده ها را به عنوان تست و بقیه را به عنوان آموزش استفاده میکنیم همچنین تابعی که در این بخش استفاده شده داده ها را بصورت پیش فرض شافل میکند. سپس با استفاده از توابع اماده مقادیر مختلف ارزیابی را بدست می اوریم. برای این کار یک دیتافریم میسازیم بطوریکه اندیس های این نام مدل ها به همراه پارامتر ها میباشد و ستون های ان دو معیار ارزیابی خواسته شده اند. که این دیتافریم بر اساس معیار دقیق مرتباً مرتب شده است نتایج مدل های مختلف به صورت زیر است مشاهده میشود که در هر دو معیار کرنل خطی بهترین عملکرد را دارد. در مورد کرنل چند جمله ای مشاهده میشود بهترین عملکرد مربوط به کرنل با درجه پنج و ضریب پنج میباشد. در مورد کرنل گاوسی بهترین عملکرد مربوط به گاما برابر `scale` میباشد که برابر معکوس تعداد ویژگی ها ضریب واریانس ان ها میباشد. در مورد کرنل سیگموید مقادیر زیادی را برای پارامتر مورد نظر امتحان کردم اما هر بار نتایج مربوط به این کرنل مشابه یکدیگر میشوند.

	accuracy score	f1 score
SVC(kernel=linear)	0.897959	0.938272
SVC(kernel=poly, degree=5, coef0=5)	0.877551	0.925000
SVC(kernel=poly, degree=12, coef0=5)	0.877551	0.918919
SVC(kernel=poly, degree=12, coef0=2)	0.877551	0.921053
SVC(kernel=poly, degree=12, coef0=1)	0.877551	0.925000
SVC(kernel=poly, degree=3, coef0=1)	0.857143	0.915663
SVC(kernel=rbf, gamma=scale)	0.857143	0.915663
SVC(kernel=poly, degree=3, coef0=5)	0.857143	0.915663
SVC(kernel=poly, degree=3, coef0=10)	0.857143	0.915663
SVC(kernel=rbf, gamma=0.01)	0.857143	0.915663
SVC(kernel=poly, degree=2, coef0=10)	0.857143	0.915663
SVC(kernel=poly, degree=5, coef0=10)	0.857143	0.909091
SVC(kernel=poly, degree=2, coef0=1)	0.857143	0.915663
SVC(kernel=poly, degree=2, coef0=5)	0.857143	0.915663
SVC(kernel=poly, degree=2, coef0=2)	0.857143	0.915663
SVC(kernel=poly, degree=3, coef0=2)	0.857143	0.915663
SVC(kernel=rbf, gamma=0.001)	0.836735	0.902439
SVC(kernel=poly, degree=12, coef0=10)	0.836735	0.891892
SVC(kernel=poly, degree=5, coef0=2)	0.836735	0.902439
SVC(kernel=poly, degree=5, coef0=1)	0.816327	0.888889
SVC(kernel=rbf, gamma=5)	0.775510	0.873563
SVC(kernel=rbf, gamma=10)	0.775510	0.873563
SVC(kernel=sigmoid, coef0=0.001)	0.775510	0.873563
SVC(kernel=sigmoid, coef0=4)	0.775510	0.873563
SVC(kernel=sigmoid, coef0=10)	0.775510	0.873563
SVC(kernel=sigmoid, coef0=15)	0.775510	0.873563
SVC(kernel=rbf, gamma=auto)	0.755102	0.860465

شکل ۱: نتایج خواسته شده



```
def model_generator(model, params):
    """a functions that generates all possible models from given parameters

    Args:
        model ([sk_learn obj]): [model from sklearn]
        params ([dict]): [parameters of model]
    """

    def model_name_generator(model, param):
        #generate name for a given model and params
        name = model.__name__ + ''.join(map(str, param.items()))
        name = name.replace(' ', '_').replace("''", "").replace(")", " ", ", ")
        return name

    def dict_product(d):
        #generate params from given ranges
        lists = list(itertools.product(*d.values()))
        dicts = [dict(zip(d.keys(), l)) for l in lists]
        return dicts

    params = dict_product(params)
    models = {model_name_generator(model, param):model(**param) for param in params}

    return models

#models is a dict that for different kernels and params has names and models
models = {
    **model_generator(SVC,
                      {'kernel': ['linear']}), #linear kernel

    **model_generator(SVC,
                      {'kernel': ['poly'],
                       'degree': range(1,6),
                       'coef0': [2**i / 2**5 for i in range(1,6)]}), #polynomial kernels

    **model_generator(SVC,
                      {'kernel': ['rbf'],
                       'gamma': ['auto', 'scale', .1, .01]}), #RBF kernels

    **model_generator(SVC,
                      {'kernel': ['sigmoid'],
                       'coef0': [2**i / 2**5 for i in range(1,6)]}) #sigmoid kernels
}
```

Listing :\



---

```
#splitting data to train and test (20% test - 80% train)
test_size = 0.25
train_x, test_x, train_y, test_y = train_test_split(df.drop(columns=[TARGET]), df[TARGET], test_size=test_size)
#defining metrics
metrics = [accuracy_score, f1_score]

#create a data frame which each row has model parameters and
#accuracy and f1 score dataframe is sorted based on accuracy
leaderboard = pd.DataFrame({
    metric.__name__: [metric(test_y, model.fit(train_x, train_y).predict(test_x)) \
        for model in models.values()] \
        for metric in metrics}, \
    index=models.keys().sort_values(metrics[0].__name__, ascending=False)
```

---

Listing :۪



## ۲.۱.۲

با توجه به نتایج مشاهده میشود برای کرنل چند جمله ای در کل با کاهش پارامترها عملکرد مدل ضعیف تر میشود. در مورد کرنل گاوی مشاهده میشود با افزایش گاما عملکرد مدل ضعیف تر میشود. در مورد کرنل سیگموید همانطور که در بخش قبل گفته شد ضرایب زیادی امتحان کردم اما هر بار نتایج این کرنل مشابه میشندند.

## ۳.۱.۲

یک روش برای این کار میتواند این باشد که با استفاده از مجموعه validation ترکیب های مختلف پارامتر ها را چک کنیم و بهترین دقت را انتخاب کنیم

## ۴.۲

## ۱.۲.۲

در این قسمت نیز برای ساختن مدل ها ازتابع generator model که در بخش قبل توضیح داده شد استفاده کردیم و همچنین برای روی مجموعه اموزش و تست دو دیتا فریم جداگانه در نظر گرفته شده است. از بیست و پنج درصد داده ها به عنوان داده تست و بقیه را به عنوان داده اموزش استفاده کردیم. کد این بخش به صورت زیر است.

```
models = {  
    **model_generator(RandomForestClassifier,  
        {'n_estimators': range(1, 6,2),  
         'max_features': range(1, 5),  
         'max_depth': range(1, 6,2)}), #for different parameters of random forest values are set 1,  
}  
  
#splitting data to train and test (20% test - 80% train)  
test_size = 0.25  
train_x, test_x, train_y, test_y = train_test_split(df.drop(columns=[TARGET]), df[TARGET], test_size=test_size)  
#defining metrics  
metrics = [accuracy_score]  
#create a data frame which each row has model parameters and accuracy. dataframe is sorted based on accuracy  
leaderboard_test = pd.DataFrame({  
    metric.__name__:[metric(test_y, model.fit(train_x, train_y).predict(test_x)) \  
        for model in models.values()] \  
        for metric in metrics}, \  
        index=models.keys().sort_values(metrics[0].__name__, ascending=False)  
  
leaderboard_train = pd.DataFrame({  
    metric.__name__:[metric(train_y, model.fit(train_x, train_y).predict(train_x)) \  
        for model in models.values()] \  
        for metric in metrics}, \  
        index=models.keys().sort_values(metrics[0].__name__, ascending=False)
```

Listing :\*



همچنین نتایج این بخش در به صورت زیر هست. مشاهده میشود که بهترین مدل با توجه به دقت روی  
مجموعه تست مدل با مشخصات زیر است

n estimators=3, max features=4, max depth=3

test accuracy:	accuracy_score
RandomForestClassifier(n_estimators=3, max_features=4, max_depth=3)	0.765625
RandomForestClassifier(n_estimators=5, max_features=4, max_depth=5)	0.760417
RandomForestClassifier(n_estimators=5, max_features=3, max_depth=5)	0.750000
RandomForestClassifier(n_estimators=5, max_features=2, max_depth=3)	0.739583
RandomForestClassifier(n_estimators=5, max_features=4, max_depth=3)	0.739583
RandomForestClassifier(n_estimators=1, max_features=2, max_depth=3)	0.739583
RandomForestClassifier(n_estimators=3, max_features=3, max_depth=3)	0.729167
RandomForestClassifier(n_estimators=5, max_features=3, max_depth=3)	0.729167
RandomForestClassifier(n_estimators=5, max_features=3, max_depth=1)	0.729167
RandomForestClassifier(n_estimators=3, max_features=1, max_depth=3)	0.729167
RandomForestClassifier(n_estimators=5, max_features=1, max_depth=3)	0.723958
RandomForestClassifier(n_estimators=3, max_features=2, max_depth=3)	0.718750
RandomForestClassifier(n_estimators=1, max_features=3, max_depth=5)	0.713542
RandomForestClassifier(n_estimators=3, max_features=4, max_depth=5)	0.703125
RandomForestClassifier(n_estimators=5, max_features=2, max_depth=5)	0.703125
RandomForestClassifier(n_estimators=3, max_features=2, max_depth=5)	0.697917
RandomForestClassifier(n_estimators=3, max_features=4, max_depth=1)	0.697917
RandomForestClassifier(n_estimators=3, max_features=3, max_depth=5)	0.692708
RandomForestClassifier(n_estimators=3, max_features=2, max_depth=1)	0.687500
RandomForestClassifier(n_estimators=5, max_features=4, max_depth=1)	0.687500
RandomForestClassifier(n_estimators=5, max_features=1, max_depth=5)	0.687500
RandomForestClassifier(n_estimators=1, max_features=3, max_depth=3)	0.682292
RandomForestClassifier(n_estimators=1, max_features=4, max_depth=3)	0.677083
RandomForestClassifier(n_estimators=1, max_features=2, max_depth=5)	0.677083
RandomForestClassifier(n_estimators=5, max_features=2, max_depth=1)	0.671875
RandomForestClassifier(n_estimators=1, max_features=1, max_depth=1)	0.661458
RandomForestClassifier(n_estimators=3, max_features=1, max_depth=5)	0.661458
RandomForestClassifier(n_estimators=3, max_features=1, max_depth=1)	0.661458
RandomForestClassifier(n_estimators=1, max_features=1, max_depth=3)	0.645833
RandomForestClassifier(n_estimators=1, max_features=4, max_depth=5)	0.645833
RandomForestClassifier(n_estimators=1, max_features=1, max_depth=5)	0.640625
RandomForestClassifier(n_estimators=3, max_features=3, max_depth=1)	0.625000
RandomForestClassifier(n_estimators=5, max_features=1, max_depth=1)	0.609375
RandomForestClassifier(n_estimators=1, max_features=4, max_depth=1)	0.609375
RandomForestClassifier(n_estimators=1, max_features=3, max_depth=1)	0.598958
RandomForestClassifier(n_estimators=1, max_features=2, max_depth=1)	0.588542

شکل ۲: نتایج خواسته شده



train accuracy:	accuracy_score
RandomForestClassifier(n_estimators=5, max_features=4, max_depth=5)	0.869792
RandomForestClassifier(n_estimators=3, max_features=4, max_depth=5)	0.855903
RandomForestClassifier(n_estimators=5, max_features=3, max_depth=5)	0.852431
RandomForestClassifier(n_estimators=3, max_features=3, max_depth=5)	0.831597
RandomForestClassifier(n_estimators=5, max_features=2, max_depth=5)	0.828125
RandomForestClassifier(n_estimators=1, max_features=2, max_depth=5)	0.810764
RandomForestClassifier(n_estimators=3, max_features=2, max_depth=5)	0.810764
RandomForestClassifier(n_estimators=1, max_features=4, max_depth=5)	0.805556
RandomForestClassifier(n_estimators=5, max_features=1, max_depth=5)	0.798611
RandomForestClassifier(n_estimators=5, max_features=2, max_depth=3)	0.798611
RandomForestClassifier(n_estimators=3, max_features=4, max_depth=3)	0.793403
RandomForestClassifier(n_estimators=5, max_features=4, max_depth=3)	0.791667
RandomForestClassifier(n_estimators=3, max_features=3, max_depth=3)	0.784722
RandomForestClassifier(n_estimators=5, max_features=3, max_depth=3)	0.781250
RandomForestClassifier(n_estimators=1, max_features=3, max_depth=5)	0.781250
RandomForestClassifier(n_estimators=3, max_features=2, max_depth=3)	0.772569
RandomForestClassifier(n_estimators=5, max_features=1, max_depth=3)	0.772569
RandomForestClassifier(n_estimators=3, max_features=1, max_depth=5)	0.763889
RandomForestClassifier(n_estimators=5, max_features=4, max_depth=1)	0.760417
RandomForestClassifier(n_estimators=1, max_features=2, max_depth=1)	0.755208
RandomForestClassifier(n_estimators=5, max_features=1, max_depth=1)	0.753472
RandomForestClassifier(n_estimators=3, max_features=2, max_depth=1)	0.753472
RandomForestClassifier(n_estimators=1, max_features=3, max_depth=1)	0.751736
RandomForestClassifier(n_estimators=1, max_features=1, max_depth=5)	0.748264
RandomForestClassifier(n_estimators=3, max_features=3, max_depth=1)	0.744792
RandomForestClassifier(n_estimators=3, max_features=1, max_depth=3)	0.736111
RandomForestClassifier(n_estimators=1, max_features=4, max_depth=3)	0.725694
RandomForestClassifier(n_estimators=1, max_features=3, max_depth=3)	0.725694
RandomForestClassifier(n_estimators=3, max_features=4, max_depth=1)	0.715278
RandomForestClassifier(n_estimators=1, max_features=2, max_depth=3)	0.711806
RandomForestClassifier(n_estimators=5, max_features=2, max_depth=1)	0.687500
RandomForestClassifier(n_estimators=1, max_features=4, max_depth=1)	0.684028
RandomForestClassifier(n_estimators=5, max_features=3, max_depth=1)	0.677083
RandomForestClassifier(n_estimators=3, max_features=1, max_depth=1)	0.664931
RandomForestClassifier(n_estimators=1, max_features=1, max_depth=3)	0.663194
RandomForestClassifier(n_estimators=1, max_features=1, max_depth=1)	0.659722

شکل ۳: نتایج خواسته شده



## ۲.۲.۲

در مورد مجموعه اموزش هر چقدر این مقادیر کمتر میشوند به طور کلی خطای افزایش میباید همچنین کاهش `n_estimators` کمترین تاثیر و کاهش `depth max` بیشترین تاثیر را روی افزایش خطای در مجموعه اموزش دارند. اما در مورد دقت در مجموعه تست مدل هایی که پارامتر هایشان مقادیر کمی دارند به دلیل بایاس بالا عملکرد خوبی ندارند همچنین اگر این پارامتر ها زیاد باشند نیز به دلیل واریانس بالا عملکرد مدل خوب نیست و در کل بنظر میرسد باید بین این متغیر ها تعادل خوبی برقرار شود تا مدل عملکرد خوبی داشته باشد.

## ۳.۲.۲

در این بخش مدل ها بصورت زیر تعریف شده اند.

```
models = {
    **model_generator(BaggingClassifier,
                      {'base_estimator': [SVC()],
                       'n_estimators': range(8,11)}),
    **model_generator(BaggingClassifier,
                      {'n_estimators': range(8,11)}),
    **model_generator(AdaBoostClassifier,
                      {}),
    **model_generator(GradientBoostingClassifier,
                      {'n_estimators': range(1, 100, 20)}),
}
```

## Listing : ۴

همچنین نتایج نیز به صورت زیر میباشند.

test accuracy:	accuracy_score
BaggingClassifier(n_estimators=9)	0.786458
GradientBoostingClassifier(n_estimators=21)	0.786458
GradientBoostingClassifier(n_estimators=81)	0.786458
BaggingClassifier(base_estimator=SVC(), n_estimators=8)	0.781250
BaggingClassifier(base_estimator=SVC(), n_estimators=9)	0.781250
BaggingClassifier(base_estimator=SVC(), n_estimators=10)	0.776042
GradientBoostingClassifier(n_estimators=61)	0.776042
AdaBoostClassifier	0.770833
GradientBoostingClassifier(n_estimators=41)	0.770833
BaggingClassifier(n_estimators=10)	0.765625
BaggingClassifier(n_estimators=8)	0.734375
GradientBoostingClassifier(n_estimators=1)	0.630208

شکل ۴: نتایج خواسته شده