

Gaussian Processes for Classification

(Incourse Assignment 01)

Course: RME-5106

Name: Ahsan Imran

Roll: 1693

July 31, 2022

1 Problem Description

Gaussian Process Classification is a Nonparametric classification method. It uses a Bayesian approach. It presumes that the underlying probability densities have a prior distribution that ensures certain smoothness properties.

The classification that best fits the observed data while also ensuring smoothness is then chosen as the final classification. This is accomplished by considering the smoothness prior and the observed classification of the training data.

For Gaussian processes, the kernel—which establishes the covariance function of the data—must be specified. The kernel regulates how samples relate to one another. The "nuisance" function or latent function is what is referred to as here. Given that the model thinks that instances that are "close" to one another have the same class label, how the examples are organized using the kernel determines how the model "perceives" the examples.

It is crucial to test various kernel functions for the model as well as various configurations for complex kernel functions. A link function that decodes the internal representation and predicts the likelihood of class membership is also necessary. It is possible to describe a binomial probability distribution for binary classification using the logistic function.

A Gaussian random variable $X \sim \mathcal{N}(\mu, \Sigma)$, where μ is the mean and Σ is the covariance matrix has the following probability density function:

$$P(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|} e^{-\frac{1}{2}((x-\mu)^\top \Sigma^{-1}(x-\mu))}$$

where $|\Sigma|$ is the determinant of Σ . The Gaussian distribution occurs very often in real world data.

There are coded as well as the descriptions of the process for two datasets from the next page.

Gaussian Processes Classification With Scikit-Learn

Dataset 1

Breast Cancer Dataset

- Classes 2
- Samples per class 213(M), 357(M)
- Samples total 569
- Dimensionality 30
- Features real, positive

▼ Tune Gaussian Processes Hyperparameters

The hyperparameters for the Gaussian Processes Classifier method must be configured for specific dataset.

Perhaps the most important hyperparameter is the kernel controlled via the “kernel” argument. The scikit-learn library provides many built-in kernels that can be used.

- RBF
- DotProduct
- Matern
- RationalQuadratic
- WhiteKernel

```
# grid search kernel for gaussian process classifier
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.gaussian_process.kernels import RBF
from sklearn.gaussian_process.kernels import DotProduct
from sklearn.gaussian_process.kernels import Matern
from sklearn.gaussian_process.kernels import RationalQuadratic
from sklearn.gaussian_process.kernels import WhiteKernel
# define dataset
X, y = load_breast_cancer(return_X_y = True)
# define model
```

```

model = GaussianProcessClassifier()
# define model evaluation method
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
print(cv)
# define grid
grid = dict()
grid['kernel'] = [1*RBF(), 1*DotProduct(), 1*Matern(), 1*RationalQuadratic(), 1*WhiteKernel()]
# define search
search = GridSearchCV(model, grid, scoring='accuracy', cv=cv, n_jobs=-1)
# perform the search
>results = search.fit(X, y)
# summarize best
print('Best Mean Accuracy: %.3f' % results.best_score_)
print('Best Config: %s' % results.best_params_)
# summarize all
means = results.cv_results_['mean_test_score']
params = results.cv_results_['params']
for mean, param in zip(means, params):
    print(">%.3f with: %r" % (mean, param))

    RepeatedStratifiedKFold(n_repeats=3, n_splits=10, random_state=1)

```

We can see the best Kernel so far is RationalQuadratic and the worst one is WhiteKernel. But it might vary with datasets.

▼ Loading Scikit-Learn Library that we will need further

```

from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RationalQuadratic
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

```

▼ Loading Breast Cancer Dataset from Scikit-Learn

```

X, y = load_breast_cancer(return_X_y = True)

```

▼ Shape of the Dataset

```
X.shape, y.shape
```

```
((569, 30), (569,))
```

▼ Splitting up the dataset-> train:test=80:20

```
# split into train test sets
```

```
X, X_test, y, y_test = train_test_split(X, y, test_size=0.2)
```

▼ Defining the model using RationalQuadratic Kernel

```
model = GaussianProcessClassifier(kernel=1**2 * RationalQuadratic(alpha=1, length_scale=1))
```

▼ Fitting the model

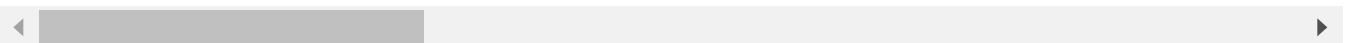
```
model.fit(X, y)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/gaussian_process/_gpc.py:472: ConvergenceWarning:
ABNORMAL_TERMINATION_IN_LNSRCH.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

```
_check_optimize_result("lbfgs", opt_res)
/usr/local/lib/python3.7/dist-packages/sklearn/gaussian_process/kernels.py:437: ConvergenceWarning:
ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/gaussian_process/kernels.py:437: ConvergenceWarning:
ConvergenceWarning,
GaussianProcessClassifier(kernel=1**2 * RationalQuadratic(alpha=1, length_scale=1))
```



▼ Make a prediction

```
yhat = model.predict(X_test)
```

yhat

```
array([1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 1])
```

▼ Precision, Recall, Fscore, Accuracy as Output:

```
precision, recall, fscore, _ = score(y_test, yhat)
accuracy = accuracy_score(y_test, yhat)
print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))

print('accuracy: {}'.format(accuracy))
```

```
precision: [0.88888889 0.92753623]
recall: [0.88888889 0.92753623]
fscore: [0.88888889 0.92753623]
accuracy: 0.9122807017543859
```

Tuning Hyperparameter such as Kernel, can give different

▼ result. E.g. if we choose WhiteKernel, it gives the worst result.

```
from sklearn.gaussian_process.kernels import WhiteKernel
model = GaussianProcessClassifier(1*WhiteKernel())
```

```
model.fit(X, y)
```

```
GaussianProcessClassifier(kernel=1*2 * WhiteKernel(noise_level=1))
```

```
yhat = model.predict(X_test)
precision, recall, fscore, _ = score(y_test, yhat)
accuracy = accuracy_score(y_test, yhat)
print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
```

```

print('fscore: {}'.format(fscore))

print('accuracy: {}'.format(accuracy))

precision: [0.39473684 0.          ]
recall: [1. 0.]
fscore: [0.56603774 0.          ]
accuracy: 0.39473684210526316
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedWarning:
    _warn_prf(average, modifier, msg_start, len(result))

```

We can see the accuracy we get using RationalQuadratic Kernel is 0.9123 and using WhiteKernel is 0.3947

Dataset 2

Wine Dataset

- Classes 3
- Samples per class [59,71,48]
- Samples total 178
- Dimensionality 13
- Features real, positive

Loading another dataset called Wine Dataset from Scikit-Learn

▼ Tune Gaussian Processes Hyperparameters

```

# grid search kernel for gaussian process classifier
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.datasets import load_wine
from sklearn.gaussian_process.kernels import RBF
from sklearn.gaussian_process.kernels import DotProduct

```

```

from sklearn.gaussian_process.kernels import DotProduct
from sklearn.gaussian_process.kernels import Matern
from sklearn.gaussian_process.kernels import RationalQuadratic
from sklearn.gaussian_process.kernels import WhiteKernel
# define dataset
X, y = load_wine(return_X_y = True)
# define model
model = GaussianProcessClassifier()
# define model evaluation method
cv = RepeatedStratifiedKfold(n_splits=10, n_repeats=3, random_state=1)
# define grid
grid = dict()
grid['kernel'] = [1*RBF(), 1*DotProduct(), 1*Matern(), 1*RationalQuadratic(), 1*WhiteKernel()]
# define search
search = GridSearchCV(model, grid, scoring='accuracy', cv=cv, n_jobs=-1)
# perform the search
results = search.fit(X, y)
# summarize best
print('Best Mean Accuracy: %.3f' % results.best_score_)
print('Best Config: %s' % results.best_params_)
# summarize all
means = results.cv_results_['mean_test_score']
params = results.cv_results_['params']
for mean, param in zip(means, params):
    print(">%.3f with: %r" % (mean, param))

Best Mean Accuracy: 0.953
Best Config: {'kernel': 1**2 * DotProduct(sigma_0=1)}
>0.932 with: {'kernel': 1**2 * RBF(length_scale=1)}
>0.953 with: {'kernel': 1**2 * DotProduct(sigma_0=1)}
>0.910 with: {'kernel': 1**2 * Matern(length_scale=1, nu=1.5)}
>0.940 with: {'kernel': 1**2 * RationalQuadratic(alpha=1, length_scale=1)}
>0.269 with: {'kernel': 1**2 * WhiteKernel(noise_level=1)}

```

As we can see that RBF, RationalQuadratic, DotProduct

- ▼ Kernel giving us better Accuracy while WhiteKernel performs worst.

```

from sklearn.datasets import load_wine

X, y = load_wine(return_X_y = True)

```

- ▼ Shape of the Dataset

```
X.shape, y.shape
```

```
((178, 13), (178,))
```

▼ Splitting up the dataset-> train:test=80:20

```
# split into train test sets
```

```
from sklearn.model_selection import train_test_split
```

```
X, X_test, y, y_test = train_test_split(X, y, test_size=0.2)
```

▼ Defining the model using DotProduct Kernel

```
from sklearn.gaussian_process.kernels import DotProduct
```

```
model = GaussianProcessClassifier(1*DotProduct())
```

▼ Fitting the model

```
model.fit(X, y)
```

```
GaussianProcessClassifier(kernel=1**2 * DotProduct(sigma_0=1))
```

▼ Make a prediction

```
yhat = model.predict(X_test)
```

```
yhat
```

```
array([1, 1, 0, 1, 2, 2, 1, 2, 1, 1, 2, 2, 1, 0, 0, 1, 1, 0, 1, 2, 2, 2,  
       0, 0, 1, 0, 0, 2, 1, 1, 1, 0, 1, 0, 2, 1])
```

▼ Precision, Recall, Fscore, Accuracy as Output:

```
precision, recall, fscore, _ = score(y_test, yhat)
```

```
accuracy = accuracy_score(y_test, yhat)
```



```

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))

print('accuracy: {}'.format(accuracy))

precision: [1.  1.  0.8]
recall: [1.          0.88888889 1.          ]
fscore: [1.          0.94117647 0.88888889]
accuracy: 0.9444444444444444

```

▼ Now we will test with RBF Kernal

```

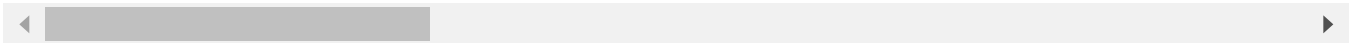
from sklearn.gaussian_process.kernels import RBF

model = GaussianProcessClassifier(1*RBF())

model.fit(X, y)

/usr/local/lib/python3.7/dist-packages/sklearn/gaussian_process/kernels.py:437: Converge
ConvergenceWarning,
GaussianProcessClassifier(kernel=1**2 * RBF(length_scale=1))

```



```

yhat = model.predict(X_test)
precision, recall, fscore, _ = score(y_test, yhat)
accuracy = accuracy_score(y_test, yhat)
print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))

print('accuracy: {}'.format(accuracy))

precision: [1.  1.  1.]
recall: [1.  1.  1.]
fscore: [1.  1.  1.]
accuracy: 1.0

```

For RBF Kernel, it gives accuracy 1 and for DotProduct 0.9444