Machine Learning
Assignment 4

Mao Xiqing (tls868)

December 13, 2020

Contents

1 Airline Revisited

Sample $S = 10000$ fight reservations and sample $S' =$ sell 100 tickets can be seen as a joint sample $S \cup S' = 10100$ passengers.

Let observed 5% no-show of sample $S = 10000$ as event A, let overbook of sample $S' = 100$ as event B, and let (9500+100) passengers show up of sample $S \cup S' = 10100$ as event C. The probability of they happened simultaneously $\mathbb{P}(AB)$ is equivalent to

$$\mathbb{P}(AB|C) \times \mathbb{P}(C). \tag{1}$$

Now event A and B are dependent. Since $\mathbb{P}(C) \in [0, 1]$, we can eliminate $\mathbb{P}(C)$ and write in this way:

$$\mathbb{P}(AB|C) \times \mathbb{P}(C) \leq \mathbb{P}(AB|C) \leq \mathbb{P}(A|C) = \left(\frac{96}{101}\right)^{100} \approx 0.00623 \tag{2}$$

2 The Growth Function and the VC-Dimension

1. The growth function of H ($m_{\mathcal{H}}(n)$) is defined as the maximal number of dichotomies it can generate on n points. Theoretically, there would be 2^n possible for n points, but sometimes these point would be linear inseparable, thus $m_{\mathcal{H}}(n) \leq 2^n$.

   A hypothesis cannot classify a point to two different results, which means it cannot decide two different dichotomies. A hypothesis also may not classify a point successfully. Therefore, M hypothesis can decide at most M dichotomies: $m_{\mathcal{H}}(n) \leq M$.

   Hence $m_{\mathcal{H}}(n) = \min\{M, 2^n\}$.

2. From the definition we know d_{VC} is the largest sample size which can be shattered by $\mathcal{H}$:

   $$d_{VC}(\mathcal{H}) = max\{n|m_{\mathcal{H}}(n) = 2^n\}. \tag{3}$$

   The break point is unknown so we can let $m_{\mathcal{H}}(n) = |\mathcal{H}| = M$, which means $M < 2^n$. Hence $d_{VC} \leq log_2(M)$.

3. We already know $m_{\mathcal{H}}(n) = \min\{M, 2^n\}$.

   If $M < 2^n$, then $m_{\mathcal{H}}(n) = M = 2$. So there is only 1 sample because it need at least two h to classify it into 0 or 1, which means $d_{VC}(\mathcal{H})$ must be 1.

   If $M > 2^n$, which means $2 > 2^n$. So there is only 1 sample because $n \in \mathbf{N}^+$. So $d_{VC}(\mathcal{H}) = 1$.

4. If $\mathcal{H}$ has a finite break point k, from Sauer's Lemma we know:

   $$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{d_{vc}(\mathcal{H})=k-1} \binom{n}{i}, \tag{4}$$

and we already proved this lemma from self-assessment:

$$\sum_{i=0}^{d} \binom{n}{i} \leq n^d + 1, \tag{5}$$

thus $m_{\mathcal{H}}(n) \leq n^{k-1} + 1$, and then

$$m_{\mathcal{H}}(2n) \leq 2n^{k-1} + 1 \leq n^{2\times k-1} + 1 = m_{\mathcal{H}}(n^2) \tag{6}$$

If $\mathcal{H}$ is non-break point, obviously $m_{\mathcal{H}}(2n) = 2^{2n} = m_{\mathcal{H}}(2n)$. Hence $m_{\mathcal{H}}(2n) \leq m_{\mathcal{H}}(2n)$.

5. A non-trivial bound on the loss means it less than 1. Thus:

$$\hat{L}(h, S) + \sqrt{\frac{8 \ln \left(2 \left((2n)^{d_{VC}} + 1\right) / \delta\right)}{n}} < 1. \tag{7}$$

To ensure left side less than 1, let:

$$\sqrt{\frac{8 \ln \left(2 \left((2n)^{d_{VC}} + 1\right) / \delta\right)}{n}} \ll 1. \tag{8}$$

Since it much less than 1, some element can be eliminate:

$$\begin{aligned} \ln \left(\frac{2\left((2n)^{d_{VC}} + 1\right)}{\delta} \right) &\ll n \\ \frac{2\left((2n)^{d_{VC}} + 1\right)}{\delta} &\ll e^n \end{aligned} \tag{9}$$

.

Since $\delta \in [0, 1]$, it can also be eliminated. Then we get the relation between $d_{VC}\mathcal{H}$ and n in the VC generalization bound:

$$\begin{aligned} (2n)^{d_{VC}} &\ll e^n \\ n^{d_{VC}} &\ll e^n \end{aligned} \tag{10}$$

6. In the Theorem 3.16, $2n^{d_{VC}(\mathcal{H})} + 1$ is come from a simple polynomial of order $d_v c$:

$$m_{\mathcal{H}}(2n) \leq \sum_{i=0}^{d_{VC}} \binom{n}{i} \leq n^{d_{VC}} + 1, \tag{11}$$

which contributes to the slack in the VC bound. And it use $m_{\mathcal{H}}(2n) = min\{M, 2^n\}$ which is also contributes to the slack because sometimes the dichotomies generated by $\mathcal{H}$ for N points may far less than 2^N. Besides, the proof of VC bound is based on union bound and Hoeffding Inequality which are already loose.

Although Theorem 3.2 also use union bound and Hoeffding Inequality, it use real n and Hypothesises size M. So Theorem 3.2 gives a tighter bound.

7. For a linear classifiers in $\mathbb{R}^{10}$, the VC-dimension would be $10+1$. From the question we know precision $\epsilon = 0.01$ and $\delta = 0.01$. From the VC generalization bound we have

$$N = \frac{8}{\epsilon^2} \ln \frac{2\left((2N)^{d_{\text{vc}}} + 1\right)}{\delta}. \tag{12}$$

Starting from $N = 10$, I get a new N around 3060110. I did a for-loop 50 times and it converge to 15609625.408065485. So we need at least 15609626 samples for this case.

8. It is obvious that when $n = 1, 2$, the points can be scattered by positive circles. And any non-linear combination of 3 points can also be scattered by positive circles, see Figure 1.

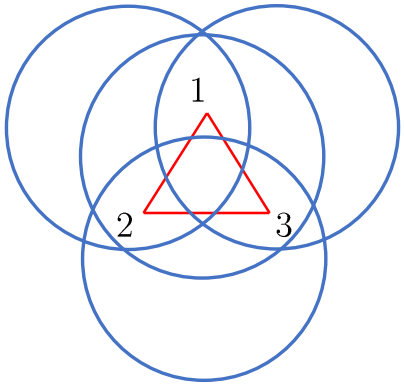

Figure 1: Example of a set of 3 points which can always be scattered by a positive circle, regardless of what the positive and negative combination of the three points are.

Thus in such case $m_{\mathcal{H}_+}(n) = 2^n$ for $n = 1, 2, 3$. We only considered $n \leq 3$, so for more points, the VC dimension is at least 3: $d_{VC_{\mathcal{H}_+}} \geq 3$.

9. For the same reason, $d_{VC_{\mathcal{H}_-}} \geq 3$. Since $\mathcal{H} = \mathcal{H}_+ \cup \mathcal{H}_+$, we have $m_{\mathcal{H}(n)} = m_{\mathcal{H}_+(n)} + m_{\mathcal{H}_-(n)} = 2^n + 2^n = 2^{n+1}$. So 4 points can be scattered by a positive and a negative circle, which means for more points, $d_{VC} \geq 4$.

3 SVMs

Source code for this question can be found in the notebook `svm.ipynb`.

3.1 Data normalization

Firstly, I split data into features and labels. Then I use `sklearn.preprocessing.scale` to normalized them, which is actually compute Z-score of data:

$$X_{norm} = \frac{x_i - \bar{X}}{\sigma}, \tag{13}$$

where σ is the standard deviation: $\sigma = sqrt(\sum_{i=1}^{n}(x_i - \bar{x})^2/n)$. Alternatively, we can use `np.mean` and `np.std` to calculate.

The mean and variance of every feature in training data are shown in Table 1.

Table 1: Mean and Var of training data

	mean of training data	**var of training data**
1	155.96	1962.81
2	204.82	9633.82
3	115.06	2093.57
4	0.01	0
5	0	0
6	0	0
7	0	0
8	0.01	0
9	0.03	0
10	0.26	0.03
11	0.01	0
12	0.02	0
13	0.02	0
14	0.04	0
15	0.02	0
16	22	16.51
17	0.49	0.01
18	0.72	0
19	-5.76	1.06
20	0.21	0.01
21	2.37	0.14
22	0.2	0.01

Note that the mean and variance of some features, for example features $5 \sim 7$, are not 0, they are just very very tiny.

All mean and variance of the transformed test data of each feature are 0 and 1, respectively .

3.2 Model selection using grid-search

I use `sklearn` to implement SVM with the RBF kernel, the range of both two hyperparameter is `10**n for n in range(-2,6)`.

For the cross validation, I use a for-loop to find the best hyperparameters, and I use `cross_val_score` to compute the score of each pair of gamma and C. Then I computed and compared the average score of each validation. The default scoring method is come form estimator, which is mean accuracy.

The parameter `cv` in `cross_val_score` decide splitting strategy. The default method in this case is `StratifiedKFold`, which selected **C = 10 and gamma = 0.1** with score 0.9084. However, if use `cv=k_fold` with default setting, the selected parameters would be **C = 1 and gamma = 0.1** with a lower score 0.8884. The different between them is that `StratifiedKFold` preserve the percentage of samples for each class while `KFold` do not.

```
cv_scores = []
hp = []
k_fold = KFold(n_splits=5) #shuffle=True, random_state=True
for i in C:
  for j in gamma:
    clf = svm.SVC(kernel='rbf', C=i, gamma=j)
    pra = [i,j]
    scores = cross_val_score(clf, x, y, cv=5)
    cv_scores.append(scores.mean())
    hp.append(pra)

# find best hyper-parameter
besthp = hp[np.argmax(cv_scores)]
print(besthp,max(cv_scores))
```

I picked **C = 10 and gamma = 0.1** with the average error 0.0916.

I use them to train model with complete training dataset. The training error is of course **0**.

I use `sklearn.metrics.accuracy_score` to compute test score: 0.9072, which is the same method used in cross validation, returning the fraction of correctly classified samples. So the test error is **0.0928**.

Alternatively, GridSearchCV is a good way to do cross validation without manually looping.

```
param=[{"kernel":["rbf"],"C":[10**n for n in range(-2,6)], "gamma": [10**
    n for n in range(-2,6)]}]
grid = GridSearchCV(svm.SVC(), param_grid=param, cv=5)
grid.fit(x,y)
print('best_params:',  grid.best_params_)
print('best_score_:', grid.best_score_)
```

3.3 Inspecting the kernel expansion

Parameter C decide the trade-off between generalization and accuracy. Larger C means more constraints on the slack variables, more weights on penalty parts and large upper bound on α. So the margin will be narrow and model won't tolerate misclassification, thus fewer bounded SV fall in the margin area.

Small C means model has more tolerance of misclassification. The margin will be wide.

When C reaches a certain level, the margin can't be smaller anymore. Now it becomes hard margin SVM. So there is no bounded SV that falls inside the margin, so the number of free SV is also fixed.

I use `dual_coef_` to get α_i, it returns $\{y_i\alpha_i\}$. Since $y_i \in [-1, +1]$, I can safely get α_i by `np.abs(model.dual_coef_)`:

```
for i in [10**n for n in range(-2,6)]:
  model_c = svm.SVC(kernel='rbf', C=i, gamma=0.1)
  model_c.fit(x,y)
  y_pred_c = model_c.predict(x_test)
  alphas = np.abs(model_c.dual_coef_)
```

If $\alpha_i < C$, meaning free SV. If $\alpha_i = C$, meaning bounded SV.

```
  free = [i , alphas[np.where(alphas < i)].shape[0]]
  bsv = [i , alphas[np.where(alphas == i)].shape[0]]
```

The output support my argument:

```
number of bounded SV: [[0.01, 39], [0.1, 38], [1, 25], [10, 1], [100, 0],
    [1000, 0], [10000, 0], [100000, 0]]
number of free SV: [[0.01, 29], [0.1, 34], [1, 40], [10, 55], [100, 54],
   [1000, 54], [10000, 54], [100000, 54]]
```