Benjamin Ye

CS/CNS/EE 156a: Learning Systems (Fall 2023)

October 23, 2023

**Homework 4**

|  |  |
| --- | --- |
| **Problem** | **Answer** |
| 1 | [d] |
| 2 | [d] |
| 3 | [c] |
| 4 | [e] |
| 5 | [b] |
| 6 | [a] |
| 7 | [b] |
| 8 | [c] |
| 9 | [b] |
| 10 |  |

**Generalization Error**

In problems 1–3, we look at generalization bounds numerically. For , use the simple approximate bound for the growth function .

1. For an with , if you want confidence that your generalization error is at most , what is the closest numerical approximation of the sample size that the VC generalization bound predicts?

**Answer: [d] 460,000**

The VC generalization bound states that for any tolerance ,

with probability . With , the bound becomes

Plugging in , , and , and solving for ,

1. There are a few bounds on the generalization error , all holding with probability at least . Fix and and plot these bounds as a function of . Which bound is the smallest for very large , say ? Note that [c] and [d] are implicit bounds in .

**Answer: [d] Devroye:**

1. For the same values of and of problem 2, but for small , say , which bound is the smallest?

**Answer: [c] Parrondo and van den Broek:**

See the next page for figures and calculations for problems 2–3.

The figure below shows the relationship between the generalization error bound as a function of the sample size , with and highlighted with vertical dotted lines.

A graph of different colored lines

Description automatically generated

The numerical value of the bounds at and are listed below:

[HW4 P2–3]

Generalization bounds for d\_vc=50 and delta=0.05:

N=10,000:

Vapnik–Chervonenkis: 0.632

Rademacher: 0.331

Parrondo–van den Broek: 0.224

Devroye: 0.215

N=5:

Vapnik–Chervonenkis: 13.828

Rademacher: 7.049

Parrondo–van den Broek: 5.101

Devroye: 5.593

(The Python 3 source code is available on the following page.)

import matplotlib as mpl

import matplotlib.pyplot as plt

import numpy as np

from scipy import optimize

mpl.rcParams.update(

    {

        "axes.labelsize": 14,

        "figure.autolayout": True,

        "figure.figsize": (4.875, 3.65625),

        "font.size": 12,

        "legend.columnspacing": 1,

        "legend.edgecolor": "1",

        "legend.framealpha": 0,

        "legend.fontsize": 12,

        "legend.handlelength": 1.25,

        "legend.labelspacing": 0.25,

        "xtick.labelsize": 12,

        "ytick.labelsize": 12,

        "text.usetex": True

    }

)

def vapnik\_chervonenkis\_bound(m\_H, N, delta):

    return np.sqrt(8 \* np.log(4 \* m\_H(2 \* N) / delta) / N)

def rademacher\_bound(m\_H, N, delta):

    return (np.sqrt(2 \* np.log(2 \* N \* m\_H(N)) / N)

            + np.sqrt(2 \* np.log(1 / delta) / N) + 1 / N)

def parrondo\_van\_den\_broek\_bound(m\_H, N, delta, \*, ub=10.0):

    return np.vectorize(

        lambda N: optimize.root\_scalar(

            lambda eps: np.sqrt((2 \* eps + np.log(6 \* m\_H(2 \* N) / delta)) / N)

                        - eps,

            bracket=(0.0, ub), method="toms748"

        ).root

    )(N)

def devroye\_bound(m\_H, N, delta, \*, ub=10.0, log=False):

    func = lambda eps, N: np.sqrt(

        (4 \* eps \* (1 + eps) + np.log(4 / delta)

         + (m\_H(N \*\* 2) if log else np.log(m\_H(N \*\* 2)))) / (2 \* N)

    ) - eps

    return np.vectorize(

        lambda N: optimize.root\_scalar(func, args=N, bracket=(0.0, ub),

                                       method="toms748").root

    )(N)

if \_\_name\_\_ == "\_\_main\_\_":

    d\_vc, delta = 50, 0.05

    m\_H = lambda N: N \*\* d\_vc

    Ns = np.arange(3, 10001, dtype=float)

    bounds = {

        "Vapnik–Chervonenkis": vapnik\_chervonenkis\_bound(m\_H, Ns, delta),

        "Rademacher": rademacher\_bound(m\_H, Ns, delta),

        "Parrondo–van den Broek": parrondo\_van\_den\_broek\_bound(m\_H, Ns, delta),

        "Devroye": devroye\_bound(lambda N: d\_vc \* np.log(N), Ns, delta, log=True)

    }

    print(f"\n[HW4 P2–3]\nGeneralization bounds for {d\_vc=} and {delta=}:")

    for N in (10000, 5):

        i = np.where(Ns == N)[0][0]

        print(f"  {N=:,}:")

        for l, b in bounds.items():

            print(f"    {l}: {b[i]:.3f}")

    \_, ax = plt.subplots()

    for l, b in bounds.items():

        ax.plot(Ns, b, label=l)

    ax.set\_yscale("log")

    ylim = ax.get\_ylim()

    ax.plot((5, 5), ylim, "k:")

    ax.plot((10000, 10000), ylim, "k:")

    ax.legend(title="Generalization bound")

    ax.set\_xlabel("$N$")

    ax.set\_xscale("log")

    ax.set\_ylabel("$\epsilon$")

    ax.set\_ylim(ylim)

    ax.text(-0.2, 0.959, " ", transform=ax.transAxes)

    plt.show()

**Bias and Variance**

Consider the case where the target function is given by and the input probability distribution is uniform on . Assume that the training set has only two examples (picked independently), and that the learning algorithm produces the hypothesis that minimizes the mean squared error on the examples.

1. Assume the learning model consists of all hypotheses of the form . What is the expected value of the hypothesis produced by the learning algorithm (expected value with respect to the data set)? Express your as , and round to two decimal digits only, the match *exactly* to one of the following answers.

**Answer: [e] None of the above**

1. What is the closest value to the bias in this case?

**Answer: [b] 0.3**

1. What is the closest value to the variance in this case?

**Answer: [a] 0.2**

See the next page for derivations, explanations, and simulations for problems 4–6.

1. Now, let’s change . Which of the following learning models has the least expected value of out-of-sample error?

**Answer: [b] Hypotheses of the form**

For [a] , the out-of-sample error is (from example 2.8 in the *Learning from Data* textbook).

For [b] , the out-of-sample error is (from problem 6).

For [c] , the out-of-sample error is (from example 2.8 in the *Learning from Data* textbook).

[d] and [e] cannot be used for a data set with a sample size of 2 since they have quadratic forms. Even if it was possible, they are expected to have much higher variances due to their increased complexity compared to hypotheses [a] through [c], and consequently, higher expected out-of-sample errors.

For a sample size of 2, the mean squared error (MSE) of a fit is given by

By minimizing the MSE with respect to , we obtain the optimal value of for the pair of points:

To answer problems 4–6, ten million pairs of and are randomly generated, the best fits are determined as outlined above, and is found by averaging the values from the best fits.

Then, the bias and variance are evaluated on a new set of ten million pairs using

A sample output is

[HW4 P4–6]

Bias and variance of h(x)=ax for f(x)=sin(pi\*x):

g(x)=1.43x, bias=0.270, var=0.236

The Python 3 source code is available below.

import numpy as np

def generate\_data(

        N, f, d=2, lb=-1.0, ub=1.0, \*, bias=False, rng=None, seed=None):

    if rng is None:

        rng = np.random.default\_rng(seed)

    x = rng.uniform(lb, ub, (N, d))

    if bias:

        x = np.hstack((np.ones((N, 1)), x))

    return x, f(x)

if \_\_name\_\_ == "\_\_main\_\_":

    n\_runs = 10\_000\_000

    x, y = generate\_data(2 \* n\_runs, lambda x: np.sin(np.pi \* x), 1)

    as\_ = (x[::2] \* y[::2] + x[1::2] \* y[1::2]) / (x[::2] \*\* 2 + x[1::2] \*\* 2)

    a\_avg = as\_.mean()

    x, y = generate\_data(2 \* n\_runs, lambda x: np.sin(np.pi \* x), 1)

    print(f"\n[HW4 P4–6]\nBias and variance of h(x)=ax for f(x)=sin(pi\*x):\n"

          f"  g(x)={a\_avg:.2f}x, bias={((a\_avg \* x - y) \*\* 2).mean():.3f}, "

          f"var={(((np.tile(as\_, (2, 1)) - a\_avg) \* x) \*\* 2).mean():.3f}")

**VC Dimension**

1. Let be an integer and assume that . What is the VC dimension of a hypothesis set whose growth function for all satisfies ? Recall that when .

**Answer: [c]**

The growth function is

When , the combination term in the right-hand side of the growth function becomes zero. By starting with and recursively multiplying by 2 for each increment in , the growth function simplifies to

Now, let’s consider the case when . For , the growth function is

Since is less than , the break point is and the VC dimension is .

1. For hypothesis sets with finite, positive VC dimensions (same input space ), some of the following bounds are correct and some are not. Which, among the correct ones, is the tightest bound (the smallest range of values) on the VC dimension of the *intersection* of the sets ? (The VC dimension of an empty set or a singleton set is taken as zero.)

**Answer: [b]**

Since the intersection of the sets can be an empty or singleton set, the lower bound must be zero.

There is some flexibility in the upper bound. By definition, a hypothesis set or intersection set with a VC dimension of can shatter data sets with points.

The loosest possible upper bound is since it will always be greater than the number of points the intersection set able to shatter.

A tighter upper bound is since it is the number of points that the most flexible hypothesis set can shatter. This would happen to be the tightest upper bound only when all hypothesis sets share the same , i.e., .

By the same logic, the tightest upper bound must be because all hypothesis sets in the intersection set should, at a minimum, be able to shatter data sets with that many points.

1. For hypothesis sets with finite, positive VC dimensions (same input space ), some of the following bounds are correct and some are not. Which, among the correct ones, is the tightest bound (the smallest range of values) on the VC dimension of the *union* of the sets ?

**Answer:**

Following the train of thought used in problem 9, the lower bound of the union set is since it is always possible to shatter that many points by taking the most flexible hypothesis set.