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 | [e] |

## Generalization Error

In Problems 1–3, we look at generalization bounds numerically. For $N > d_{\text{VC}}$, use the simple approximate bound $N^{d_{\text{VC}}}$ for the growth function $m_{\mathcal{H}}(N)$.

1. For an $\mathcal{H}$ with $d_{\text{VC}} = 10$, if you want 95% confidence that your generalization error is at most 0.05, 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 $\delta > 0$,

   $$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

   with probability $\geq 1 - \delta$. With $m_{\mathcal{H}}(N) \leq N^{d_{\text{VC}}}$, the bound becomes

   $$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N} \ln \frac{4(2N)^{d_{\text{VC}}}}{\delta}}$$

   Plugging in $d_{\text{VC}} = 10$, $E_{\text{out}}(g) - E_{\text{in}}(g) = 0.05$, and $\delta = 0.05$, and solving for $N$,

   $$0.05 \leq \sqrt{\frac{8}{N} \ln \frac{4(2N)^{10}}{0.05}} \longrightarrow N \geq 452{,}957$$

2. There are a few bounds on the generalization error $\epsilon$, all holding with probability at least $1 - \delta$. Fix $d_{\text{VC}} = 50$ and $\delta = 0.05$ and plot these bounds as a function of $N$. Which bound is the smallest for very large $N$, say $N = 10{,}000$? Note that [c] and [d] are implicit bounds in $\epsilon$.

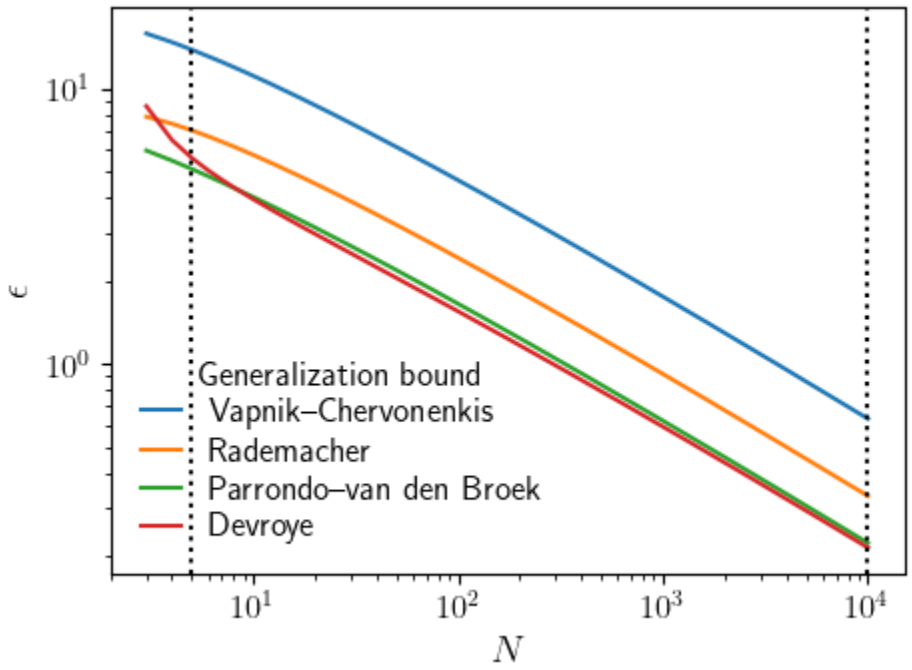
   **Answer: [d] Devroye: $\epsilon \leq \sqrt{\frac{1}{2N}\left(4\epsilon(1+\epsilon) + \ln \frac{4m_{\mathcal{H}}(N^2)}{\delta}\right)}$**

3. For the same values of $d_{\text{VC}}$ and $\delta$ of Problem 2, but for small $N$, say $N = 5$, which bound is the smallest?

   **Answer: [c] Parrondo and van den Broek: $\epsilon \leq \sqrt{\frac{1}{N}\left(2\epsilon + \ln \frac{6m_{\mathcal{H}}(2N)}{\delta}\right)}$**

(The figure, sample program output, and Python 3 source code are on the following pages. ⟶)

## Figure




## Sample program output

```
[Homework 4 Problems 2–3]
Generalization bounds:
      N  Vapnik–Chervonenkis  Rademacher  Parrondo–van den Broek  Devroye
10000.0              0.632175    0.331309                0.223698 0.215228
    5.0             13.828161    7.048777                5.101362 5.593126
```

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

### Python 3 source code

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
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):
    return np.vectorize(
        lambda N: optimize.root_scalar(
            lambda eps, N: np.sqrt(
                (4 * eps * (1 + eps) + np.log(4 / delta) + np.log(m_H(N ** 2)))
                / (2 * N)
            ) - eps,
            args=N,
            bracket=(0.0, ub),
            method="toms748"
        ).root
    )(N)

if __name__ == "__main__":
    d_vc = 50
    delta = 0.05
    m_H = lambda N: N ** d_vc
    Ns = np.arange(3, 10_001, dtype=np.longdouble)
    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(m_H, Ns, delta)
    }
    df = pd.DataFrame(columns=["N"] + [b for b in bounds.keys()])
    _, ax = plt.subplots()
    for label, bound in bounds.items():
        ax.plot(Ns, bound, label=label)
    ax.set_yscale("log")
    ylim = ax.get_ylim()
    for N in (10_000, 5):
        df.loc[len(df)] = N, *[bound[np.where(Ns == N)[0][0]]
                               for bound in bounds.values()]
        ax.plot((N, N), ylim, "k:")
    ax.legend(title="Generalization bound")
    ax.set_xlabel("$N$")
    ax.set_xscale("log")
    ax.set_ylabel("$\epsilon$")
    ax.set_ylim(ylim)
    plt.show()
    print("\n[Homework 4 Problems 2–3]\nGeneralization bounds:\n",
          df.to_string(index=False), sep="")
```

**Bias and Variance**

Consider the case where the target function $f : [-1, 1] \rightarrow \mathbb{R}$ is given by $f(x) = \sin(\pi x)$ and the input probability distribution is uniform on $[-1, 1]$. 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.

4. Assume the learning model consists of all hypotheses of the form $h(x) = ax$. What is the expected value $\bar{g}(x)$ of the hypothesis produced by the learning algorithm (expected value with respect to the data set)? Express your $\bar{g}(x)$ as $\hat{a}x$, and round $\hat{a}$ to two decimal digits only, the match *exactly* to one of the following answers.

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

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

   **Answer: [b] 0.3**

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

   **Answer: [a] 0.2**

7. Now, let's change $\mathcal{H}$. Which of the following learning models has the least expected value of out-of-sample error?

   **Answer: [b] Hypotheses of the form $h(x) = ax$**

   Simulation results show that

   - for [a] $h(x) = b$, the out-of-sample error is $\mathbb{E}_{\mathcal{D}}[E_{\text{out}}] = \text{bias} + \text{var} = 0.500 + 0.250 = 0.750$ (which matches bias + var = 0.50 + 0.25 = 0.75 from Example 2.8 in the *Learning from Data* textbook),
   - for [b] $h(x) = ax$, the out-of-sample error is $\mathbb{E}_{\mathcal{D}}[E_{\text{out}}] = \text{bias} + \text{var} = 0.271 + 0.236 = 0.507$ (from Problem 6),
   - for [c] $h(x) = ax + b$, the out-of-sample error is $\mathbb{E}_{\mathcal{D}}[E_{\text{out}}] = \text{bias} + \text{var} = 0.207 + 1.677 = 1.884$ (which agrees with bias + var = 0.21 + 1.69 = 1.90 from Example 2.8 in the *Learning from Data* textbook), and
   - for [d] $h(x) = ax^2$ and [e] $h(x) = ax^2 + b$, the out-of-sample errors are multiple orders of magnitude greater than those for hypotheses [a] through [c]. This is not surprising since the variances are expected to be much higher due to the increased complexity of the forms of the hypotheses compared to those for [a] through [c].

(The derivations, sample program output, and Python 3 source code are on the following pages. ⟶)

## Derivations

For the hypothesis with form $h(x) = ax$ and a sample size of 2, the mean squared error (MSE) is

$$\text{MSE} = \frac{1}{2}\sum_{i=1}^{2}\big(h(\mathbf{x}_i) - f(\mathbf{x}_i)\big)^2 = \frac{1}{2}\sum_{i=1}^{2}(ax_i - y_i)^2 = \frac{1}{2}[(ax_1 - y_1)^2 + (ax_2 - y_2)^2]$$

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

$$\frac{\partial}{\partial a}\text{MSE} = ax_1^2 - x_1y_1 + ax_2^2 - x_2y_2 = 0 \longrightarrow a = \frac{x_1y_1 + x_2y_2}{x_1^2 + x_2^2}$$

In the simulations, ten million pairs of $(x_1,\, y_1)$ and $(x_2,\, y_2)$ are randomly generated, the optimized $a$ values are determined using the formula above, and $\hat{a}$ is found by averaging the $a$ values.

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

$$\mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\big(\bar{g}(\mathbf{x}_n) - f(\mathbf{x}_n)\big)^2 = \frac{1}{N}\sum_{n=1}^{N}(\hat{a}x_n - \sin(\pi x_n))^2$$

$$\mathbb{E}_{\mathbf{x}}[\text{var}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\Big(g^{(\mathcal{D})}(\mathbf{x}_n) - \bar{g}(\mathbf{x}_n)\Big)^2 = \frac{1}{N}\sum_{n=1}^{N}\big((a_n - \hat{a})x_n\big)^2$$

For the four other hypotheses with linear and quadratic forms in Problem 7, their MSE, optimal values for $a$ (and $b$), bias, and variance can be calculated in a fashion like that above and are given by

- [a] $h(x) = b$

$$\text{MSE} = \frac{1}{2}\sum_{i=1}^{2}(b - y_i)^2 = \frac{1}{2}[(b - y_1)^2 + (b - y_2)^2]$$

$$\frac{\partial}{\partial a}\text{MSE} = 2b - y_1 - y_2 = 0 \longrightarrow b = \frac{y_1 + y_2}{2}$$

$$\mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\big(\hat{b} - \sin(\pi x_n)\big)^2, \quad \mathbb{E}_{\mathbf{x}}[\text{var}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\big(b_n - \hat{b}\big)^2$$

- [c] $h(x) = ax + b$

$$\text{MSE} = \frac{1}{2}\sum_{i=1}^{2}(ax_i + b - y_i)^2 = \frac{1}{2}[(ax_1 + b - y_1)^2 + (ax_2 + b - y_2)^2]$$

$$\left.\begin{aligned} \frac{\partial}{\partial a}\text{MSE} = 0 \longrightarrow a &= \frac{x_1(y_1 - b) + x_2(y_2 - b)}{x_1^2 + x_2^2} \\ \frac{\partial}{\partial b}\text{MSE} = 0 \longrightarrow b &= \frac{y_1 + y_2 - a(x_1 + x_2)}{2} \end{aligned}\right\} \longrightarrow a = \frac{y_1 - y_2}{x_1 - x_2},\ \ b = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2}$$

$$\mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{a}x_n + \hat{b} - \sin(\pi x_n)\right)^2,\ \ \mathbb{E}_{\mathbf{x}}[\text{var}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\left(a_n x_n + b_n - \hat{a}x_n - \hat{b}\right)^2$$

- [d] $h(x) = ax^2$

$$\text{MSE} = \frac{1}{2}\sum_{i=1}^{2}\left(ax_i^2 - y_i\right)^2 = \frac{1}{2}[(ax_1^2 - y_1)^2 + (ax_2^2 - y_2)^2]$$

$$\frac{\partial}{\partial a}\text{MSE} = 2ax_1(ax_1^2 - y_1) + 2ax_2(ax_2^2 - y_2) = 0 \longrightarrow a = \frac{x_1 y_1 + x_2 y_2}{x_1^3 + x_2^3}$$

$$\mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}(\hat{a}x_n^2 - \sin(\pi x_n))^2,\ \ \mathbb{E}_{\mathbf{x}}[\text{var}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\left((a_n - \hat{a})x_n^2\right)^2$$

- [e] $h(x) = ax^2 + b$

$$\text{MSE} = \frac{1}{2}\sum_{i=1}^{2}\left(ax_i^2 + b - y_i\right)^2 = \frac{1}{2}[(ax_1^2 + b - y_1)^2 + (ax_2^2 + b - y_2)^2]$$

$$\left.\begin{aligned} \frac{\partial}{\partial a}\text{MSE} = 0 \longrightarrow a &= \frac{x_1(y_1 - b) + x_2(y_2 - b)}{x_1^3 + x_2^3} \\ \frac{\partial}{\partial b}\text{MSE} = 0 \longrightarrow b &= \frac{y_1 + y_2 - a(x_1^2 + x_2^2)}{2} \end{aligned}\right\} \longrightarrow a = \frac{y_1 - y_2}{x_1^2 - x_2^2},\ \ b = \frac{x_1^2 y_2 - x_2^2 y_1}{x_1^2 - x_2^2}$$

$$\mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{a}x_n^2 + \hat{b} - \sin(\pi x_n)\right)^2,\ \ \mathbb{E}_{\mathbf{x}}[\text{var}(\mathbf{x})] = \frac{1}{N}\sum_{n=1}^{N}\left((a_n - \hat{a})x_n^2 + b_n - \hat{b}\right)^2$$

(The sample program output and Python 3 source code are on the following pages. ⟶)

### Sample program output

```
[Homework 4 Problems 4–7]
Learning algorithms for f(x)=sin(pi*x):
choice   h(x)          g(x)      bias     variance
   [a]      b          0.00  0.499970 2.499281e-01
   [b]     ax         1.43x  0.270684 2.365877e-01
   [c]   ax+b    0.78x+0.00  0.206688 1.676365e+00
   [d]   ax^2      14.63x^2 43.338501 2.449373e+07
   [e] ax^2+b 21.95x^2-0.68 87.382128 4.026515e+07
```

Note that the bias and variance for [d] and [e] fluctuate wildly but are consistently much greater than those for [a] through [c].

### Python 3 source code

```
import numpy as np
import pandas as pd

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
    f = lambda x: np.sin(np.pi * x)
    hs = {  # h(x): ((a(x_in, y_in), b(x_in, y_in)),
            #        bias(x_out, y_out, a_avg, b_avg),
            #        var(x_out, a, a_avg, b, b_avg),
            #        fmt(a_avg, b_avg))
        "b": (
            lambda x, y: (None, (y[::2] + y[1::2]) / 2),
            lambda xt, yt, ah, bh: ((bh - yt) ** 2).mean(),
            lambda xt, a, ah, b, bh: ((np.tile(b, (2, 1)) - bh) ** 2).mean(),
            lambda ah, bh: f"{bh:.2f}"
        ),
        "ax": (
            lambda x, y: (
                (x[::2] * y[::2] + x[1::2] * y[1::2])
                / (x[::2] ** 2 + x[1::2] ** 2),
                None
            ),
            lambda xt, yt, ah, bh: ((ah * xt - yt) ** 2).mean(),


            lambda xt, a, ah, b, bh: (
                ((np.tile(a, (2, 1)) - ah) * xt) ** 2
            ).mean(),
            lambda ah, bh: f"{ah:.2f}x"
```

```
    ),
    "ax+b": (
        lambda x, y: (
            (y[::2] - y[1::2]) / (x[::2] - x[1::2]),
            (x[::2] * y[1::2] - x[1::2] * y[::2]) / (x[::2] - x[1::2])
        ),
        lambda xt, yt, ah, bh: ((ah * xt + bh - yt) ** 2).mean(),
        lambda xt, a, ah, b, bh: (
            ((np.tile(a, (2, 1)) - ah) * xt + np.tile(b, (2, 1)) - bh) ** 2
        ).mean(),
        lambda ah, bh: f"{ah:.2f}x{'+' if bh >= 0 else ''}{bh:.2f}"
    ),
    "ax^2": (
        lambda x, y: (
            (x[::2] * y[::2] + x[1::2] * y[1::2])
            / (x[::2] ** 3 + x[1::2] ** 3),
            None
        ),
        lambda xt, yt, ah, bh: ((ah * xt ** 2 - yt) ** 2).mean(),
        lambda xt, a, ah, b, bh: (
            ((np.tile(a, (2, 1)) - ah) * xt ** 2) ** 2
        ).mean(),
        lambda ah, bh: f"{ah:.2f}x^2"
    ),
    "ax^2+b": (
        lambda x, y: (
            (y[::2] - y[1::2]) / (x[::2] ** 2 - x[1::2] ** 2),
            (x[::2] ** 2 * y[1::2] - x[1::2] ** 2 * y[::2])
            / (x[::2] ** 2 - x[1::2] ** 2)
        ),
        lambda xt, yt, ah, bh: ((ah * xt ** 2 + bh - yt) ** 2).mean(),
        lambda xt, a, ah, b, bh: (
            ((np.tile(a, (2, 1)) - ah) * xt ** 2
             + np.tile(b, (2, 1)) - bh) ** 2
        ).mean(),
        lambda ah, bh: f"{ah:.2f}x^2{'+' if bh >= 0 else ''}{bh:.2f}"
    )
}
x_train, y_train = generate_data(2 * n_runs, f, 1)
x_test, y_test = generate_data(2 * n_runs, f, 1)
df = pd.DataFrame(columns=["choice", "h(x)", "g(x)", "bias", "variance"])
for i, (h, (f_ab, f_bias, f_var, fmt)) in enumerate(hs.items()):
    as_, bs = f_ab(x_train, y_train)
    a_avg = None if as_ is None else as_.mean()
    b_avg = None if bs is None else bs.mean()
    bias = f_bias(x_test, y_test, a_avg, b_avg)
    var = f_var(x_test, as_, a_avg, bs, b_avg)
    df.loc[len(df)] = f"[{chr(97 + i)}]", h, fmt(a_avg, b_avg), bias, var
print("\n[Homework 4 Problems 4–7]\n"
      "Learning algorithms for f(x)=sin(pi*x):\n",
      df.to_string(index=False), sep="")
```

**VC Dimension**

8. Let $q \geq 1$ be an integer and assume that $m_{\mathcal{H}}(1) = 2$. What is the VC dimension of a hypothesis set whose growth function for all $N \geq 1$ satisfies $m_{\mathcal{H}}(N+1) = 2m_{\mathcal{H}}(N) - C_q^N$? Recall that $C_m^M = 0$ when $m > M$.

   **Answer: [c] $q$**

   The growth function is

$$m_{\mathcal{H}}(N+1) = 2m_{\mathcal{H}}(N) - \binom{N}{q}$$

   When $N \leq q$, the combination term in the right-hand side of the growth function becomes zero. By starting with $N = 1$ and recursively multiplying by 2 for each increment in $N$, the growth function simplifies to

$$m_{\mathcal{H}}(N \leq q) = 2^N$$

   Now, let's consider the case when $N > q$. For $N = q + 1$, the growth function is

$$m_{\mathcal{H}}(q+1) = 2m_{\mathcal{H}}(q) - \binom{q}{q} = 2^{q+1} - 1$$

   Since $m_{\mathcal{H}}(q+1)$ is less than $m_{\mathcal{H}}(N \leq q)$, the break point is $k = q + 1$ and the VC dimension is $d_{\text{VC}} = k - 1 = q$.

9. For hypothesis sets $\mathcal{H}_1$, $\mathcal{H}_2$, …, $\mathcal{H}_K$ with finite, positive VC dimensions $d_{\text{VC}}(\mathcal{H}_k)$ (same input space $\mathcal{X}$), 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 $d_{\text{VC}}\left(\cap_{k=1}^K \mathcal{H}_k\right)$? (The VC dimension of an empty set or a singleton set is taken as zero.)

   **Answer: [b] $0 \leq d_{\text{VC}}\left(\cap_{k=1}^K \mathcal{H}_k\right) \leq \min\{d_{\text{VC}}(\mathcal{H}_k)\}_{k=1}^K$**

   Since the intersection of the sets $\mathcal{H}_k$ 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 $d_{\text{VC}}$ can shatter data sets with $d_{\text{VC}}$ points.

   The loosest possible upper bound is $\sum_{k=1}^K d_{\text{VC}}(\mathcal{H}_k)$ since it will always be greater than the number of points the intersection set able to shatter, even if $d_{\text{VC}}(\mathcal{H}_k) = 0$ for all hypothesis sets.

   A tighter upper bound is $\max\{d_{\text{VC}}(\mathcal{H}_k)\}_{k=1}^K$ 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 $d_{\text{VC}}$, i.e., $d_{\text{VC}}(\mathcal{H}_1) = d_{\text{VC}}(\mathcal{H}_2) = \cdots = d_{\text{VC}}(\mathcal{H}_K)$.

   By the same logic, the tightest upper bound must be $\min\{d_{\text{VC}}(\mathcal{H}_k)\}_{k=1}^K$ for the general case because all hypothesis sets in the intersection set should, at a minimum, be able to shatter data sets with that many points.

10. For hypothesis sets $\mathcal{H}_1$, $\mathcal{H}_2$, …, $\mathcal{H}_K$ with finite, positive VC dimensions $d_{\text{VC}}(\mathcal{H}_k)$ (same input space $\mathcal{X}$), 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 $d_{\text{VC}}\left(\cup_{k=1}^{K} \mathcal{H}_k\right)$?

**Answer: [e]** $\boldsymbol{\max\{d_{\text{VC}}(\mathcal{H}_k)\}_{k=1}^{K} \leq d_{\text{VC}}\left(\cup_{k=1}^{K} \mathcal{H}_k\right) \leq K - 1 + \sum_{k=1}^{K} d_{\text{VC}}(\mathcal{H}_k)}$

Following the train of thought used in Problem 9, the lower bound of the union set is $\max\{d_{\text{VC}}(\mathcal{H}_k)\}_{k=1}^{K}$ since it is always possible to shatter that many points by taking the most "flexible" hypothesis set, which has the highest VC dimension.

One approach to find the upper bound is to determine the break point (and consequently, the VC dimension) using the growth function of the union set of two hypothesis sets and work recursively from that/use mathematical induction.

For any two hypothesis sets $\mathcal{H}_i$ and $\mathcal{H}_j$ in the union set, the growth function is bounded by

$$m_{\mathcal{H}_i \cup \mathcal{H}_j}(N) \leq m_{\mathcal{H}_i}(N) + m_{\mathcal{H}_j}(N) \leq \sum_{k=0}^{d_{\text{VC}}(\mathcal{H}_i)} \binom{N}{k} + \sum_{k=0}^{d_{\text{VC}}(\mathcal{H}_j)} \binom{N}{k}$$

since it cannot be more than the sum of the individual growth functions (limiting case where two subsets of a data set with VC dimensions of $d_{\text{VC}}(\mathcal{H}_i)$ and $d_{\text{VC}}(\mathcal{H}_j)$, respectively, are classified correctly exactly and only by the two hypothesis sets).

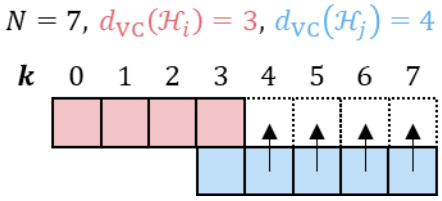
Rewriting the second term using the equality $C_k^n = C_{n-k}^n$ (from a Pascal's triangle perspective) and applying a change of variables $k' = N - k$,

$$m_{\mathcal{H}_i \cup \mathcal{H}_j}(N) \leq \sum_{k=0}^{d_{\text{VC}}(\mathcal{H}_i)} \binom{N}{k} + \sum_{k=0}^{d_{\text{VC}}(\mathcal{H}_j)} \binom{N}{N-k} = \sum_{k=0}^{d_{\text{VC}}(\mathcal{H}_i)} \binom{N}{k} + \sum_{k'=N-d_{\text{VC}}(\mathcal{H}_j)}^{N} \binom{N}{k'}$$

Then, the break point can be determined by finding the $N$ that gives $m_{\mathcal{H}_i \cup \mathcal{H}_j}(N) \leq 2^N - f(N)$, where $f$ is some arbitrary positive function.

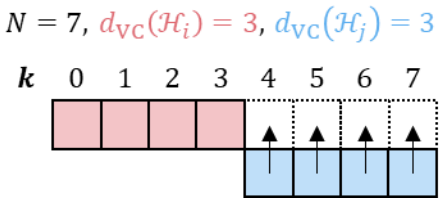
The first summation becomes equal to $2^N$ when the $C_k^N$ contributions from $k = d_{\text{VC}}(\mathcal{H}_i) + 1$ to $k = N$ is added to it.

- When $N \leq d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j)$, the second summation has more than enough terms to "transfer" to the first summation to change the upper bound of the latter to $k = N$. After this "move", the second summation still has remaining terms, making it a positive term (thus not satisfying the conditions for the $-f(N)$ term). An example with $N = d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j)$ is shown in the schematic below, where each box is a combination term, and the top and bottom rows represent the first and second summations.

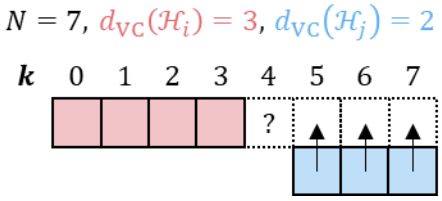



  If $d_{\text{VC}}(\mathcal{H}_i)$ or $d_{\text{VC}}(\mathcal{H}_j)$ increases, there will be more boxes on the second row left over and the second summation stays a positive term. Therefore, $N < d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j)$ also holds.

- When $N = d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j) + 1$, the second summation has just enough terms that, when combined with the first summation, gives $\sum_{k=0}^{N} C_k^N = 2^N$.




- When $N \geq d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j) + 2$, the second summation does not have enough terms to "complete" the first summation. An example is shown below for $N = d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j) + 2$ only, but the same scenario arises when $N > d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j) + 2$.




  As such, the missing terms come from outside the second summation and must be subtracted away in the end, i.e., $m_{\mathcal{H}_i \cup \mathcal{H}_j}(N) \leq \sum_{k=0}^{N} C_k^N - f(N)$, where $f(N) = \sum_{k=\alpha}^{\beta} C_k^N$, and $\alpha$ and $\beta$ are the smaller and larger, respectively, of $\{N - d_{\text{VC}}(\mathcal{H}_j),\ N - d_{\text{VC}}(\mathcal{H}_i) - 1\}$. Therefore, the break point is $k = d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j) + 2$, and the corresponding VC dimension is $d_{\text{VC}}(\mathcal{H}_i \cup \mathcal{H}_j) = k - 1 = d_{\text{VC}}(\mathcal{H}_i) + d_{\text{VC}}(\mathcal{H}_j) + 1$.

This means that the VC dimension of the union set of two hypothesis sets is bounded by

$$d_{\mathrm{VC}}\left(\mathcal{H}_i \cup \mathcal{H}_j\right) \leq d_{\mathrm{VC}}(\mathcal{H}_i) + d_{\mathrm{VC}}\left(\mathcal{H}_j\right) + 1$$

From the logic and schematics on the previous page, it is obvious that for each additional hypothesis set $\mathcal{H}_k$, the number of boxes (and consequently, the VC dimension) increases by $d_{\mathrm{VC}}(\mathcal{H}_k) + 1$.

For example, the upper bound on the VC dimension for a union set with three hypothesis sets is

$$d_{\mathrm{VC}}\left(\cup_{k=0}^{3} \mathcal{H}_k\right) \leq d_{\mathrm{VC}}(\mathcal{H}_1 \cup \mathcal{H}_2) + d_{\mathrm{VC}}(\mathcal{H}_3) + 1 = d_{\mathrm{VC}}(\mathcal{H}_1) + d_{\mathrm{VC}}(\mathcal{H}_2) + d_{\mathrm{VC}}(\mathcal{H}_3) + 2$$

By generalizing for a union set with $K$ hypothesis sets, the upper bound on the VC dimension can be shown to be

$$d_{\mathrm{VC}}\left(\cup_{k=0}^{K} \mathcal{H}_k\right) \leq \sum_{k=0}^{K} d_{\mathrm{VC}}(\mathcal{H}_k) + K - 1$$